

DOCUMENT RESUME

ED 346 816

IR 014 350

TITLE An Introduction to Computer Science (Secondary Grades). Volume 1.

INSTITUTION New York City Board of Education, Brooklyn, N.Y.

REPORT NO ISBN-88315-570-2

PUB DATE 84

NOTE 336p.; For volume 2, see IR 014 351.

AVAILABLE FROM New York City Board of Education, Curriculum Unit, Room 310 P.S., Neck Road & East 22nd Street, Brooklyn, NY 11229.

PUB TYPE Guides - Classroom Use - Teaching Guides (For Teacher) (052)

EDRS PRICE MF01 Plus Postage. PC Not Available from EDRS.

DESCRIPTORS *Computer Literacy; *Computer Science Education; Curriculum Guides; High Schools; *Microcomputers; *Programming; Word Processing

ABSTRACT

The first volume of a two-volume computer science curriculum guide, reflecting the technology of the mid-1980s, presents 61 lessons on advanced computer literacy topics for high school students. The majority of the lessons focus on advanced programming skills in BASIC. Other topics covered include word processing, printers, document management, text storage, and using a word processor to modify program and text files. The appendixes present lessons on 11 programming review topics. Each lesson includes the following sections: performance objectives, vocabulary, motivation, aim, development, application, summary, and exercises. (MES)

* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *

ED346816

IR014330

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION
CENTER (ERIC)

* This document has been reproduced as
received from the person or organization
originating it.

Minor changes have been made to improve
reproduction quality.

• Points of view or opinions stated in this docu-
ment do not necessarily represent official
OERI position or policy.

Computer Science

"PERMISSION TO REPRODUCE THIS
MATERIAL IN MICROFICHE ONLY
HAS BEEN GRANTED BY

Dorothy Klein

TO THE EDUCATIONAL RESOURCES
INFORMATION CENTER (ERIC)."

VOLUME I

of Curriculum and Instruction



Office of Curriculum Development and Support

AN INTRODUCTION TO
Computer Science
(SECONDARY GRADES)

VOLUME I

NEW YORK CITY BOARD OF EDUCATION

Division of Curriculum and Instruction



Office of Curriculum Development and Support

EXPERIMENTAL EDITION

New York City Board of Education

James F. Regan, *President*
Miguel O. Martinez, *Vice President*

Amelia Ashe
Joseph G. Barkan
Stephen R. Franse
Irene Impellizzeri
Majorie A. Lewis

Nathan Quinones, *Chancellor*

Charles I. Schonhaut, *Deputy Chancellor*

Louise Latty, *Chief Executive for Instruction*

Charlotte Frank, *Executive Director*
Division of Curriculum and Instruction

Copyright 1984 by the Board of Education of the City of New York.

Application for permission to reprint any section of this material should be made to the Chancellor, 110 Livingston Street, Brooklyn, N.Y. 11201. Reprints of any section of this material shall carry the line, "Reprinted from An Introduction to Computer Science (Secondary Grades), Volume I, by permission of the Board of Education of the City of New York."

ISBN 88315-570-2

CURRIC. NO. 00-6090-81

New York City Public Schools should order additional copies of this publication from the Bureau of Supplies.

Other persons and institutions may obtain copies of this publication from the Publications Sales Office, Room 136, 110 Livingston St., Brooklyn, NY 11201. (See current catalog for price.)

FOREWORD

In 1983 the Division of Curriculum and Instruction published computer literacy curriculum guides for both elementary and secondary schools. That project was the first step in a comprehensive plan for computer education in New York City. Computer technology is new and is changing daily. We as educators are faced with a common challenge: how to use this exciting and powerful technology in a way that will meet the changing needs of the students and will continue to accommodate changes as they occur. The challenge is a great one, but the potential for these tools is even greater.

We know that computers can ignite the imagination and extend the limits of human learning. They enable their users to view the world in new ways. When students interact with computers they take charge of their own learning and learn more through this interaction. The problem of time on task disappears as students work with computers to solve problems in various ways. In short, computers are tools used by humans for human endeavors.

This two-volume computer science curriculum guide is the next step in the New York City plan for computer education. It has been designed as a support document for teachers whose students have already completed earlier computer courses in the sequence. We hope that this guide will aid teachers and students in keeping pace with the challenges of our information age.



Charlotte Frank
Executive Director
Division of Curriculum and Instruction

ACKNOWLEDGMENTS

An Introduction to Computer Science (Secondary Grades), Volume I is a project of the Division of Curriculum and Instruction, Charlotte Frank, Executive Director.

Morris Freedman, Director of the Office of Curriculum Development and Support, and Irwin Kaufman, Director of Technology, provided overall supervision of this project.

Michael Ryan, Director of the Computer and Information Sciences Unit, and Joseph Eaione, Assistant Director, provided overall coordination.

Stephen Radin of Community School District 22 was the principal author and Stephen Radin and Joseph Eaione were responsible for the editing and sequencing of this guide.

The dedicated professionals listed below were involved in the writing and review of this document.

Nicholas Androu - Elizabeth Blackwell JHS 210, Ozone Park, New York
Laney Dahl - Curtis High School, Staten Island, New York
Elaine Fitzpatrick - William A. Morris IS 61, Staten Island, New York
Saundra Newman - Stuyvesant High School, New York, New York
Baruch Rapaport - A. Cunningham JHS 234, Brooklyn, New York
Richard Rothenberg - Stuyvesant High School, New York, New York

The material contained in the final version of this guide was reviewed by Gerald Elgarten, Director of Mathematics (Interim Acting); Randy Ross, Principal, Newtown High School; and the following Assistant Principals (Supervision - Mathematics): Pierre Fabinski, Christopher Columbus H.S.; Joel Friedberg, Herbert Lehman H.S.; Marc Licht, Prospect Heights H.S.; Demosthenes Lyriotakis, Manhattan Voc.-Tech. H.S.; Mary Gahn, John Bowne H.S.; Alice Farkouh, Curtis H.S.; Robert Frankel, Tottenville H.S..

The manuscript was prepared for publication by the Editorial and Production Unit, Alaire Mitchell, Unit Head.

Graphics supervision: Maureen Maguire
Word processing: Gloria Solomon
Cover: Kent Beaty

TABLE OF CONTENTS

Topic 1	
OVERVIEW	1
Topic 2	
INTERNAL DOCUMENTATION FOR THE PROGRAMMER (REMs)	5
Topic 3	
STRING AND NUMERIC VARIABLES	9
Topic 4	
RANDOM NUMBERS	13
Topic 5	
READ/DATA	17
Topic 6	
NESTED FOR/NEXT LOOPS	22
Topic 7	
SUBROUTINES	28
Topic 8	
ALGORITHMS, MODULES	31
Topic 9	
ARRAYS: SUBSCRIPTED VARIABLES	33
Topic 10	
ARRAY VARIABLES	39
Topic 11	
PROGRAMS WITH SUBROUTINES	44
Topic 12	
MORE COMPLICATED SUBROUTINES	46
Topic 13	
INTRODUCTION TO MID\$, RIGHT\$, LEFT\$ STRING MANIPULATION FUNCTIONS	49
Topic 14	
USING MID\$, RIGHT\$, AND LEFT\$ TO SEARCH FOR A CHARACTER IN A STRING AND THEN TO REVERSE AND OTHERWISE INTERCHANGE A STRING'S PARTS	53
Topic 15	
WHEN TO USE A SUBROUTINE	55
Topic 16	
SPARING USE OF VARIABLES	57
Topic 17	
WHAT MAKES A PROGRAM USER-FRIENDLY OR PROGRAMMER-FRIENDLY?	60

Topic 18	
INPUTs, INKEY\$ (OR GET\$) THAT ALLOW EVALUATIONS OF UPPERCASE AND LOWERCASE RESPONSES	62
Topic 19	
USE OF LEFT\$ TO EVALUATE PARTIAL RESPONSES	65
Topic 20	
USE THAT ALL MULTIPLE SPELLINGS OF USER RESPONSES	67
Topic 21	
CREATING PROGRAMS THAT ALLOW FOR USER ERROR IN ALL INPUTs	69
Topic 22	
ERROR TRAPPING	73
Topic 23	
MENUS FOR USER-FRIENDLY CONTROL	76
Topic 24	
INTERNAL DOCUMENTATION FOR THE USER (PRINTs) STRESSING UNAMBIGUOUS LANGUAGE	79
Topic 25	
EXTERNAL DOCUMENTATION IN THE FORM OF PRINTED DESCRIPTIVE MATERIAL	82
Topic 26	
IMPROVING USER-FRIENDLINESS BY EXTENDING INPUT SCOPE TO ALLOW TOLERANCE FOR USER RESPONSES THAT ARE CLOSE BUT NOT EXACT	84
Topic 27	
EASILY UNDERSTOOD DESCRIPTIONS (PART I)	87
Topic 28	
EASILY UNDERSTOOD DESCRIPTIONS (PART II)	95
Topic 29	
SCREEN FORMATTING	102
Topic 30	
NEAT AND ORDERLY MESSAGES	106
Topic 31	
USING STILL GRAPHICS TO CLARIFY SCREENS	109
Topic 32	
WRITING CONCISE PROGRAMS	113
Topic 33	
MACHINE LANGUAGE AND WALK THROUGH MEMORY (COMPUTERS OTHER THAN APPLE)	118
Topic 34	
MACHINE LANGUAGE AND WALK THROUGH MEMORY (APPLE COMPUTERS)	122

Topic 35	
DEVELOPING MOVING GRAPHICS USING THE PRINT@ STATEMENT	130
Topic 36	
MOVING SCREEN PATTERNS USING HTAB AND VTAB	134
Topic 37	
MOVING GRAPHICS TECHNIQUES USING PRINT@, SET, AND RESET	137
Topic 38	
MOVING BLOCK GRAPHICS WITH HTAB AND VTAB	141
Topic 39	
RAM, ROM, AND OTHER MEMORY	144
Topic 40	
COMPUTER MEMORY ORGANIZATION AND I/O BUFFERS	152
Topic 41	
INTRODUCTION TO CASSETTE AND DISK I/O	157
Topic 42	
SEQUENTIAL DISK FILE STRUCTURE	163
Topic 43	
CAI/CMI PROGRAMS	170
Topic 44	
USING STR\$ AND VAL IN A PROGRAM	176
Topic 45	
USING ASC AND CHR\$ WITH FILE MANAGEMENT PROGRAMS	182
Topic 46	
DATA SEARCHES	187
Topic 47	
LINEAR SEARCH	193
Topic 48	
BINARY SEARCH	195
Topic 49	
BUBBLE SORT: INTRODUCTION	198
Topic 50	
BUBBLE SORT: CONTINUED	200

Topic 51	
SELECTION SORT	203
Topic 52	
EDITING A PROGRAM: LINE EDITING (RADIO SHACK, APPLE MBASIC, ETC.)	206
Topic 53	
EDITING A PROGRAM: FULL SCREEN EDITING (COMMODORE)	217
Topic 54	
DOCUMENT PREPARATION	224
Topic 55	
EDITING TEXT ON A WORD PROCESSOR	229
Topic 56	
THE SEARCH/FIND FUNCTION ON A WORD PROCESSOR	233
Topic 57	
PRINTERS	237
Topic 58	
DOCUMENT MANAGEMENT	241
Topic 59	
UNDERSTANDING TEXT STORAGE	
BY EXPLORING SEVERAL WAYS OF CREATING ASCII FILES	246
Topic 60	
USING A WORD PROCESSOR TO MODIFY BASIC PROGRAMS	251
Topic 61	
USING A WORD PROCESSOR TO MODIFY BASIC TEXT FILES	257
Appendix A1	
TYPES OF VARIABLES: REVIEW	265
Appendix A2	
THE PRINT STATEMENT: REVIEW	268
Appendix A3	
FURTHER USES OF THE PRINT STATEMENT: REVIEW	273
Appendix A4	
INPUT STATEMENTS: REVIEW	278
Appendix A5	
FLOWCHARTING: REVIEW	284
Appendix A6	
READ/DATA STATEMENTS (PART I): REVIEW	289
Appendix A7	
READ/DATA STATEMENTS (PART II): REVIEW	294

Appendix A8	
BRANCHING (PART I): REVIEW	300
Appendix A9	
BRANCHING (PART II): REVIEW	305
Appendix A10	
FOR/NEXT LOOPS: REVIEW	310
Appendix A11	
VIDEO DISPLAY SCREENS	315

NOTES ON THE USE OF THIS TWO-VOLUME GUIDE

This book is the first part of a two-volume series. The series was designed to provide teachers with enough support material to enable them to teach an advanced course in computer literacy. It is intended for teachers whose classes have completed all of the work presented in our earlier programming guides. Volume II extends the objectives of Volume I.

The topics in these books are each designed to encompass an entire unit of instruction rather than a specific time period. Some may require only a few minutes to present while others will take a few days to complete.

The average high school class studying this material for three lecture and two lab periods per week, should be able to successfully complete all of the topics in approximately three semesters.

The appendix section in volume I contains eleven review topics. It was designed to assist teachers whose students have not studied computer literacy for more than one year, or as a review for more advanced students. The appendix sections included in volume II include material to help teachers meet the needs of bilingual youngsters, special education students, and those with needs for more advanced material.

Since each classroom situation is different, it is the staff of each school that must determine how this guide can be used most effectively. In fact, the individual lessons may be used in their current format or may be adapted to meet the particular needs of individual groups of students.

INTRODUCTION

The use of computers in classrooms all over the United States has been little short of spectacular in recent years. Surveys taken late in 1983 indicate that more than 85% of all schools were using microcomputers as instructional tools. The use of microcomputers in education has grown at an exponential rate since their introduction in the late 1970's. The optimum method for harnessing the power of these tools, however, is still a subject of considerable debate. The real issue is that there is no consensus on what educational computing should be.

Recent reports show that administrators and teachers feel comfortable discussing reading, math and other more familiar curricular problems. They have had years of experience and research to reinforce any decisions made to meet these challenges. Computer education is a very different matter. The technology is new and it is changing daily as is the support structure surrounding all of the new equipment. Software, texts, self-help guides and a vast array of peripheral equipment are regularly changing the face of educational computing. For one of the few times in recent history, educators across the country are faced with a common challenge: how to use this exciting and powerful technology in a way which will meet the changing needs of students and society within the constraints of existing computer technology and which will continue to accomodate these changes as they occur. The challenge is a great one, but the potential for these tools is even greater.

We know that computers ignite the imagination and can extend the limits of human learning. They enable their users to view the world in new ways. Research indicates that when students interact with computers they take charge of thier own learning and learn more through this interaction. The problem of time on task disappears as students work with computers to solve problems in various ways. In short, computers are tools used by humans for human endeavors. The days of computers being used solely for "number crunching" are long past. Computers are now more accurately viewed as information processors, a much more factually correct and less limiting view.

The New York City Board of Education has devised a plan for implementing computer education in all of its schools and for all of its students so as to best meet this great challenge. After numerous meetings with representatives of all levels of our instructional system and related community agencies, the master plan was devised. This two-volume computer science curriculum guide is but a small part of that plan. It has been designed as a support document for teachers whose students have already completed computer courses which appear earlier in the plan's sequence. We find ourselves at the dawn of a new era in education. We fervently approach the prospect and reach out to its promise.

It is the policy of the New York City Board of Education not to discriminate on the basis of race, creed, national origin, age, handicapping condition, or sex in its educational programs, activities, and employment policies, as required by law. Inquiries regarding compliance with appropriate laws may be directed to Dr. Nicholas Aiello, Local Equal Opportunity Coordinator, Division of Personnel, 65 Court Street, Brooklyn, New York 11201; Ms. Lynette V. Tucker, Director, Office of Equal Opportunity, 110 Livingston Street, Brooklyn, New York 11201; or to the Director of the Office for Civil Rights, Department of Health and Human Services, Washington, D.C. 20201.

Topic 1

OVERVIEW

Performance Objectives: Students will be able to:

- . identify the four major tasks in data processing
- . explain the functions of each of the major components of data processing
- . identify and explain the functions of computer peripherals
- . use computer peripherals for storage and display of information
- . identify the three types of computers, according to size

Vocabulary: New: cassette recorder, daisy wheel printer, disk drive, dot matrix printer, floppy disk, modem

Review: central processing unit, CRT, data processing input, keyboard, output, peripheral, keyboard, storage

Motivation:

- . Each year the Internal Revenue Service receives millions of forms, called income tax returns. These forms are usually accompanied by a check for taxes owed. Some show that the government owes a refund to the taxpayer. When the IRS receives these forms, what is done with the information on them? (input)
- . How is the information supplied by the taxpayers used?
- . When the information is used to perform a task such as an audit, what is the using of information called? (processing)
- . The information is later retrieved for further use. What procedure allows the information to be held within the computer for use at a later time? (storage)
- . Finally, some taxpayers are sent refund checks. What is the procedure of sending something out called? (output)

Aim: What computer equipment is needed to execute the four main tasks of data processing?

Development:

1. Identify the four main data processing operations.
 - . Inputting: the computer receives information
 - . Processing: the computer uses information to perform a task
 - . Storing: the computer stores information within its memory
 - . Outputting: the computer displays the information requested

2. How might the Internal Revenue Service use these data processing functions to perform its job?
 - . Inputting: The IRS feeds the information on the tax returns into the computer.
 - . Processing: The IRS takes the information supplied by the taxpayer and determines the correct tax liability.
 - . Storing: The IRS stores all information for use later, usually up to seven years.
 - . Outputting: the IRS pays a refund to those who paid too much in taxes and sends a letter including a bill to those who owe more.
3. Show that within a computer these four activities occur for specific reasons. Illustrate the purpose of each of these activities.
 - . Inputting: Data and instructions are given to the computer.
 - . Processing: The computer interprets the data and calculations.
 - . Storing: The computer stores the data, instructions, and processed information.
 - . Outputting: The computer transmits the results to an external outlet.
4. Storage occurs within the memory of the computer. Describe the two parts of memory, RAM and ROM.
 - . ROM (Read Only Memory): This section of memory stores information that is not lost when the power is turned off. ROM contains the control program (BASIC compiler) and therefore stores all instructions that are permanently a part of the computer.
 - . RAM (Random Access Memory): This section of memory is obtained by the programming information supplied by the user. This memory can be changed and all information stored within RAM is lost when the power is turned off.
5. Suppose you have written a program and wish to store it for later use. How could you do this?
 - . What type of equipment is available to provide permanent storage of programs and data for the computer?
6. Introduce a peripheral as a device which permits input or output of programs and data onto a permanent storage device.
 - . What types of peripherals exist today? (cassette tape recorder, disk drive, printer, modem)

7. What is the function of each of these devices?

Answers:

A cassette tape recorder is an input/output device that permits you to save information on a cassette or to load information from a cassette into a computer.

A disk drive is an input/output device that permits you to save information on a floppy disk or to load information and data from a disk into a computer.

A printer is an output device that permits you to make a hard copy of a program or of the results of running a program.

A modem is an input/output device that permits computers to talk to each other. By use of two modems and a phone line, one computer is able to communicate with another computer via a telephone line.

Applications:

1. How can we use a cassette recorder to save a program from memory and how may we use it to load a program? (Review the procedures for loading and saving programs and data on cassette tape.)
2. How may we use a disk drive to save a program from memory? How may we use a disk drive to load a program into memory? (Review the procedures for loading and saving programs and data on a floppy disk.)
3. What types of printers are available now? (daisy wheel, dot matrix, ink jet, laser) How are they similiar? How are they different?

Summary:

1. What are the four data processing operations?
2. What types of memory exist within the computer?
3. What peripherals may be used to save the information stored within the computer's memory?

Assignments:

1. Write a program which will compute the first 200 odd numbers. Store this program on cassette or floppy disk, whichever is available.
2. Explain how a cassette recorder and a floppy disk drive are similiar. How are they different?
3. Explain how a program can be printed out by means of a printer. How may a program be executed and the output sent to a printer?

4. Write a program that will print your initials like this:

tttttttttt	eeeeeeeeeeee
tt	ee
tt	ee
tt	ee
tt	eeeeee
tt	ee
tt	ee
tt	ee
tt	eeeeeeeeeeee

Then make a hard copy of the output with a printer.

Topic 2

INTERNAL DOCUMENTATION FOR THE PROGRAMMER (REMs)

Performance Objectives: Students will be able to:

- . define REM
- . write programs that include internal documentation
- . list four uses for REMarks within a program
- . list three reasons why REMs should be included in a program

Vocabulary: Review: REM, documentation

Motivation:

- . What does this program do?

```
10 CLS      (OR HOME OR SHIFT CLR/HOME)
30 FOR Q = 1 TO 4
40 READ N$
50 PRINT N$
60 FOR I = 1 TO 250
70 NEXT I
90 NEXT Q
95 CLS      (OR HOME OR SHIFT CLR/HOME)
100 RESTORE
110 GOTO 30
140 DATA "I", "THINK", "YOU'RE", "GREAT!"
```

- . Why are REMarks important? (A programmer can insert explanatory comments throughout the program that have no effect on the computer's operation.)
- . What are the basic rules for using REMs?
 - Each REM statement must have a line number.
 - If the REMark is longer than one line, a new line number followed by REM is needed.
 - A REM cannot precede a non-REMark statement on a multiple statement line.
- . Who is the only person who sees REM statements? (the programmer)
- . What is the purpose of a REM statement? (It serves as a heading to include the programmer's name, the date, and the purpose of the lab and to delineate major subroutines.)
- . Why is it that as programs become more complicated, almost every line may have a REM? (The REM helps explain the purpose of a variable or the function of the BASIC statement. This is especially important if we have to modify a program that we wrote months or years ago. Also, programs often outlast programmers. A new programmer may have to recode lines as company policies change. A good programmer makes a program that is easy to reread.)
- . Internal documentation will help us convert machine-dependent programs to comparable programs that can be used by an incompatible machine.

Aim: How do we use REMS to document a program internally?

Development:

1. What are appropriate REMS for the Motivation program?

Acceptable answer:

```
5 REM **** This program flashes a message one word ***
6 REM ****          at a time on a TRS-80          ***
10 CLS      (OR HOME OR SHIFT CLR/HOME) :REM clears the screen
30 FOR Q = 1 TO 4          :REM Loop for reading in data
40 READ N$
50 PRINT N$
60 FOR I = 1 TO 250        :REM time delay
70 NEXT I
90 NEXT Q
95 CLS      (OR HOME OR SHIFT CLR/HOME) :REM clears the screen
100 RESTORE                :REM resets DATA pointer
110 GOTO 30                 :REM starts message again
```

2. Now conversion or modification of the program for another machine becomes easier. If you have a PET or APPLE, how would you change lines 10 and 95 so the program will work on your machine? (Apple - HOME, Pet - SHIFT CLR/HOME)
3. Modify the program with a message of your choice. Line 30 will have to be changed depending on the number of words and the DATA will have to be changed.
4. Just as PRINT, alone, lets us skip a line during program execution, REM alone allows us to skip a line during a list to make the program easier to read. REMS are added to separate subroutines or modules within a program from each other.

```
10 REM*** Reading-In Routine      ***
20 FOR X = 1 TO 10
30 READ N(X)
40 NEXT X
50 REM
60 REM SUM = 0          : REM Initializing Variable
65 REM
70 REM *** Summation Loop ***
80 FOR X = 1 TO 10
90 SUM = SUM + N(X)
100 NEXT X
110 REM
120 REM *** Calculating the average ***
130 AV = SUM/10
140 PRINT TAB(10) "The average is ";AV
150 REM
```


160 DATA 3,57,95,43,66,75,12,95,34,27
999 END

Applications:

1. Design a marquee with the heading and purpose of the lab enclosed in a rectangle of stars. The end product should look like this:

```
*****  
*                                     *  
*   Last, First                      DATE   *  
*   Computer Class                  Teacher *  
*                                     *  
*               LAB # or Assign #      *  
*                                     *  
*   The purpose of this lab is to      *  
*                                     *  
*****
```

Have this heading and other utility routines, such as line drawing and creating a row of stars, in one program. Then, when you start a new program, you can load in your utility programs and only have to modify the headings for the new assignment.

2. The slugging percentage of a baseball hitter is the comparison of total bases to at-bats, written in decimal form to the nearest thousandth. Write a program that will read in a player's name, number of singles, doubles, triples, home runs, and times at bat and will calculate the player's slugging percentage. Make sure that the program is thoroughly documented.

Summary:

1. What four ways are REMs used in a program? (to define the author and the purpose of a program, to define the purpose of each variable, to define the purpose of a BASIC statement, and to skip a line in a LISTing)
2. Why are REMs important? (They allow us to modify a program, understand it and make it more readable.)

Assignments:

1. A baseball pitcher's earned-run average is calculated by multiplying 9 by the number of earned runs allowed and then dividing the result by the number of innings that pitcher worked. Write a program that will read in the name of a pitcher, the number of earned runs allowed, and the number of innings pitched and will calculate the ERA. Document the program so it is easily readable.
2. Write a program that can be used to calculate the averages of all of the students in your class. Include enough documentation so that it can be easily used next year.

3. Write a program that could be used by a bank to see if a person is a customer and therefore able to cash a check. Write a program with a dummy set of account numbers, names, and balances (10 would be reasonable). Use an INPUT statement to allow the teller to type in the account number, name, and amount of the check that the customer tenders. Your program should check the account number and verify the name and ascertain whether there is enough money in the account to cash the check. Remember to include documentation.
4. Write a program that will calculate the charge for the following grocery items purchased by a customer: 4 cans of tomato paste at 22 cents a can, 5 bottles of soda at 30 cents a bottle, 2 loaves of bread at \$1.05 each, 2 dozen eggs at \$1.40 a dozen.

Assignment for More Advanced Students:

1. Write a sort program that will read 40 names and homerooms and give the user a choice of sorting either alphabetically or numerically. Supply full documentation.

Topic 3

STRING AND NUMERIC VARIABLES

Performance Objectives: Students will be able to:

- . define and write numeric and string variables
- . compare and contrast numeric and string variables
- . define relational operators with respect to string and numeric variables
- . write a program using conditionals and string variables

Vocabulary: Review: relational operators, alphanumeric, string variables, numeric variables

Motivation:

- . Examine the following program. Identify any errors within the program.

```
10 PRINT "NAME", , "AVERAGE"
20 READ A, B, C, D, E
30 AV = B + C + D + E / 4
40 PRINT A, , AV
50 GOTO 20
60 DATA Walters, 84,76,39,54, Edwards, 64,20,30,88,
   FIELDS, 27,69,99,85, Marks, 90,30,100,89
70 END
```

Errors:

```
20 READ A$, B, C, D, E
30 AV= (B + C + D + E)/4
```

- . What type of variable is A\$? (a string variable)
- . Why must we use the string variable A\$, not A?
- . What type of variable is B? (a numeric variable)

Aim: How can string variables and numeric variables be used within a program?

Development:

1. What is a numeric variable? (A numeric variable names a memory location for a number or an arithmetic expression. It initializes a value for a particular variable.)
 - . In BASIC, a numeric variable can be a single letter of the alphabet or a letter followed by another letter or digit (0 to 9) (for example, A, BX, N9, R1).
 - . Many computers recognize only the first two characters of a variable. SUM1 and SUM2 would both be treated as the same memory location, SU.
2. Besides using numeric variables for assignment statements, such as D = 4,

how else may numeric variables be used? (Numeric variables may be used in mathematical operations, such as $R = R + 1$, $Z = AM + B3$. Numeric variables can also be used with relational operators such as $IF X > 4$ THEN...)

3. Write a program that will permit a user to INPUT a number and then will output the information as to whether the number is divisible by 3 or 9. (Hint: If a number is divisible by 9 then it is divisible by 3.)

```

10 INPUT "ENTER A NUMBER TO BE TESTED FOR DIVISIBILITY BY 3 OR 9"; X
20 D = X/9
30 IF D = INT(D) THEN 70
40 D = X/3
50 IF D = INT(D) THEN 80
60 PRINT X "IS NOT DIVISIBLE BY 3 OR 9": GOTO 90
70 PRINT X "IS DIVISIBLE BY 3 AND 9": GOTO 90
80 PRINT X "IS DIVISIBLE BY 3": GOTO 90
90 GOTO 10

```

4. Suppose we wish to write a program that will test words to determine if the words begin with the letters A, B, or C. We will use READ/DATA statements instead of an INPUT statement so that we can obtain a list of all words which begin with an A, B, or C.

- . What type of variable must we use in the READ statement? (a string variable)

5. A string variable is a variable used to represent a series of alphanumeric characters. In BASIC, a string variable begins with a single letter or two letters, followed by a \$ (for example, A\$, B\$, C\$).

6. Examine the following program. Lines 20 and 30 are missing:

```

10 READ W$
20
30
40 DATA GREEN, APPLE, RED, CARROT, BLUE, DANCE, PRIVATE, LIGHT, XXX
50 PRINT W$
60 GOTO 10
70 END

```

- . Look at the DATA statement of line 40. Why is the XXX used? (as a flag to indicate to the program the end of the line of DATA)

- . What is the purpose of this flag? (The use of the flag will prevent an "OUT OF DATA" error from occurring.)

- . How might I write line 20 as the test to determine if I have reached the flag in the DATA statement? (20 IF W\$ = "XXX" THEN 70)

7. Remember that the alphanumeric value assigned to the string is always enclosed by quotation marks. As a result $W\$ = "XXX"$, not $W\$ = XXX$.

8. How might we compose line 30, the line which will test the words to determine if they begin with an A, B, or C?

. How might this be accomplished in a single programming statement?

. We know that computers may compare numeric variables. It is also possible to instruct a computer to compare alphanumeric or string variables in the same fashion, using a relational operator.

. What are the relational operators?

Answer:

> greater than
< less than
>= greater than or equal to
<= less than or equal to
= equal to
< > not equal to

. As a result we may then have programming statements such as IF A\$ < G\$ THEN X\$ = "HELLO"

. How does a computer handle alphanumeric data with relational operators? (The computer assigns values to the strings and compares these values.)

. treaty < trust, too > taste, birds < > robins, avid = avid

Note: Uppercase and lowercase characters have different numerical equivalent values within the computer. (Generally, uppercase < lowercase.)

9. If we wish to determine whether a word begins with the letters A, B, or C we may assume that the word must begin with a letter less than what letter? (D)

. How may we then create the programming statement for line 30 to accomplish this task? We wish to compare the word to D, and if it is less than D we wish to print the word. Otherwise we wish to go to the next word in the DATA statement. (30 IF W\$ < "D" THEN 60 ELSE 70) Remember, if your machine has no ELSE statement you must replace line 30 with two lines.

10. Write a program that will read the following DATA and will output only those words whose first two letters are from AA to AK.

Data: ARK, APPLE, ABLE, ACCEPT, BABY, AFTER, CAT, ABSENT

Acceptable answer:

```
10 READ M$
20 IF M$ = "END" THEN 70
30 IF M$ < "AL" THEN 50 ELSE GOTO 60
40 DATA ARK, APPLE, ABLE, ACCEPT, BABY, AFTER, CAT, ABSENT, END
50 PRINT M$
70 END
```


Applications:

1. Write a program that will alphabetically output the words hen, crystal, rose, and guitar.

Summary:

1. Compare and contrast numeric and string variables. How are they similar? How are they different?
2. How does a computer compare numeric variables?
3. How does a computer compare string variables?

Assignments:

1. Dr. Chandler is inviting all his patients to a seminar on cavity prevention. However, since he has a large group of patients, he wishes to set up the appointments for this meeting on three different days. He will hold the seminar for patients whose last names begin with the letters A to LE on Monday, June 14, those whose last names begin with LF to R on Monday, June 21, and those whose last names begin with S to Z on Monday, June 28. Write a program that will print a written message to each person, telling when to attend the seminar.

Patients: Mr. Davis, Mrs. Redding, Ms. Harris, Mr. Marone,
Ms. Fitzpatrick, Miss Cousins, Mr. Perez, Mrs. Lewis, Mr. Greenberg,
Mrs. Colon

2. Write a program that will print out an alphabetized list of the following words:

daughter, dance, dugout, double
3. Develop a program that will INPUT a list of names and print out those names in alphabetical order. The list should include at least 10 names.
4. Create a program that will take the following data and arrange the items according to zip code.

Data: Harmon, 278 Reed St., 12301; Ferry, 677 Park Ave., 10314;
Marcy, 56 Lowe Terr., 61322; Able, 48 Cherry Lane, 12301;
Edwards, 14 Seven Dr., 06604; Johnson, 16 Karr St., 61322

Topic 4 RANDOM NUMBERS

Performance Objectives: Students will be able to:

- . define and explain how the INTeGer statement can be used within a program
- . define and explain how the RaNDom statement can be used within a program
- . write a program using the INTeGer and RaNDom statements

Vocabulary: Review: INT, RND

Motivation:

- . If you were tossing a die how many possible outcomes could there be? (6)
- . What are the possible outcomes? (1, 2, 3, 4, 5, or 6)
- . Would one outcome be more likely to occur than another, provided you had a "legal" die? (no)
- . Since each outcome would be equally likely, what may we say about the outcomes? (They occur randomly.)

Aim: How can we program the computer to choose randomly among numbers, letters, words, etc.?

Development:

1. Let's write a program to toss a die. The computer will determine what the toss is and whether the outcome is even or odd. What statement may be used to select randomly a number from 1 to 6? (RND)
2. What does the RaNDom statement do? (It tells the computer to select a number.)
 - . How can we limit the range of the machine's selection of numbers? How can we limit the choice to the numbers from 1 to 6? (The largest number to be chosen must be stipulated.)
 - . How can we write a command to have the computer randomly select a number from 1 to 6? (RND (6) on some machines and INT (RND(2)*6)+1 on others)
3. What numbers might the computer choose, based on the following RND commands?

RND (10) or INT(RND(2)*10)+1 (1,2,3,4,5,6,7,8,9, or 10)
RND (2) or INT(RND(2)*2)+1 (1,2)
RND (100) or INT(RND(2)*100)+1 (1,2,3,...100)

. What commands might be used to let the computer randomly select:

a number less than 5

a number less than 200
a number less than 1

4. After the computer has selected a number between 1 and 6, how can it tell if the number is even or odd?

- . How may we decide if a number is even or odd? (See if it is divisible by two.)
- . When is a number not divisible by two? (when, after dividing it by 2, it has a remainder)
- . How can a computer decide if a divisor has a remainder? (when it is not a whole number)
- . What computer statement will truncate any number leaving only its whole number portion? (INT)
 - INT(X) converts X to the largest integer value which is not greater than X.
- . What is the output for each of the following commands?

```
PRINT INT (6)           (6)
PRINT INT (2.5)         (2)
PRINT INT (15.20233)    (15)
PRINT INT (-4)          (-4)
PRINT INT (-2.43)       (-3)
PRINT INT (-63.759)     (-64)
```

5. Let us now write the program:

```
10 X = RND(6)
20 PRINT"THE NUMBER IS";X
30 Y = X/2
40 IF INT(Y) = Y THEN PRINT "EVEN" ELSE PRINT"ODD"
50 END
```

Note: Remember, if your machine will not allow ELSE you must replace one IF statement with two. In this case you would need:

```
40 IF INT(Y) = Y THEN PRINT "EVEN"
45 IF INT(Y) <> Y THEN PRINT "ODD"
```

Applications:

1. Write a program that will randomly select a number from 1 to 100 and will then ask the user to guess the number. The computer must then tell the user if the guess is correct, too high, or too low. It must then permit the user to guess again, until the user guesses the correct number.

Acceptable answer:

```
10 X = RND(100)
```

```

-----
10 X = INT(RND(2)*100)+1    or
-----
20 INPUT "I AM THINKING OF A NUMBER BETWEEN 1 AND 100. WHAT IS THE
NUMBER?"; Y
30 IF X = Y THEN 80
40 IF X < Y THEN 60
50 PRINT "YOUR GUESS IS TOO LOW":GOTO 70
60 PRINT"YOUR GUESS IS TOO HIGH":GOTO 70
70 GOTO 20
80 PRINT"CONGRATULATIONS! YOUR ANSWER IS CORRECT"
90 END

```

2. Write a program that will simulate the rolling of two dice and will record the outcome for twenty rolls.

Acceptable answer:

```

10 PRINT " DIE #1", "DIE #2"
20 FOR I = 1 to 20
30 X = RND (6)
40 Y = RND (6)
-----
or 30 X = INT(RND(2)*6)+1
40 Y = INT(RND(2)*6)+1
-----
50 PRINT X, Y
60 NEXT I

```

Summary:

1. What happens when you apply the INT statement to a positive number? (The computer prints only the whole number before the decimal point.) What happens when you apply the INT statement to a negative number? (The computer prints only the next lowest integer.)
2. How may a computer generate numbers randomly?

Assignments:

1. Write a program that will randomly select two numbers less than or equal to 1000 and then permit the user to find the sum of those two numbers and tell the user if the response was correct.
2. Write a program that will use the following list of persons and will randomly select three of the names.

Data: ED JONES, MARY STARKEY, AL ORTIZ, EDNA TEXT, LESLIE HORN, FRED GRANT, MARIA OVERTON, THOMAS LEE

3. Write a program that will ask for a student's name and test grades and then print out the person's name and grades, to the nearest integer equal to or less than the average.
4. Write a program that will tell how many times a 3 is rolled when a die is rolled twenty times.

Topic 5 READ/DATA

Performance Objectives: Students will be able to:

- . identify the need for multiple use of a DATA statement
- . write a program using the same data for more than one task
- . identify the ABSolute value of a number
- . use the ABS statement within a computer program

Vocabulary: New: ABS

Review: DATA, INT, READ, RESTORE

Motivation:

- . Mrs. Johnson, an economist, made a study of the average salary of computer technicians, using data provided by the ABC Computer Corporation. She wished to write a program that would store the technicians' names as data and then compute their average salary based on the data she recorded. The data provided by the ABC COMPUTER CORPORATION was as follows:

<u>Employee</u>	<u>Yearly salary</u>
SUSAN ADAMS	\$16450
HENRY DUZANT	14500
BARUCH JONES	18023
MARISSA MORRIS	23900
LOUISE TUPPER	17750
HOM-HONG WEI	25955

- . How would we store the data within a program? (in a DATA statement)
- . How could an average salary be arrived at? (Divide the sum of the yearly salaries by the number of salaries.)
- . Let's now write the program:

```
10 REM* READ IN THE DATA*
20 READ A$,B
30 IF B=-1 THEN 90
40 REM**FIND THE SUM OF THE SALARIES
50 AV=AV+B
60 X=X+1
70 GOTO 20
80 REM*** FIND THE AVERAGE SALARY
90 AV=AV/X
100 PRINT"THE AVERAGE SALARY IS" AV
110 DATA SUSAN ADAMS,16450,HENRY DUZANT,14500
120 DATA BARUCH JONES,18023,MARISSA MORRIS,23900
130 DATA LOUISE TUPPER,17750,HOM HONG WEI,25955
140 DATA END,-1
150 END
```

- . Suppose after writing the program Mrs. Johnson found that she must know how many employees earned more than the average salary and how many employees earned less than the average salary. Would it be necessary to add new DATA statements or would it be possible to use the existing DATA statements again?

Aim: How can DATA be READ more than once within a program without rewriting the DATA each time it is to be READ?

Development:

1. How does a computer read DATA that is stored in a program? (The computer reads one item of information for each READ variable.)
 - . What happens when the computer is asked to READ and there is no DATA to READ? (An "OUT OF DATA" error is indicated by the computer.)
 - . How may this problem be avoided? (Use a flag in the program to show where the last DATA entry is READ. At that point a decision must be made as to where the computer proceeds once all DATA is READ.)
 - . Was a flag used within the program? If so, where? (DATA END,-1)
2. Once this flag is encountered, where does the program proceed? (to line 90, where the average is computed)
 - . We are able to determine the average salary on line 100. What occurs after this point? (The program is ended.)
3. If we added a new READ statement what would happen? (The computer would print an "OUT OF DATA" error.)
 - . What might we do to cause the computer to reread the DATA statements and complete our task of comparing the average salary to the salary of each employee? (Create a statement that would cause the computer to READ all the DATA once again.)
 - . Review the RESTORE statement. The RESTORE statement causes the computer to READ all DATA from the first DATA entry on, even though it has read through the DATA before. The RESTORE statement may be used at any point within the reading of DATA or after all DATA has been READ.
4. Now let us complete our program:

```

150 RESTORE
160 READ A$,B
170 IF B=-1 THEN 220
180 IF AV<B THEN H=H+1
190 IF AV>B THEN L=L+1
200 IF AV=B THEN E=E+1
210 GOTO 160
220 PRINT"NUMBER OF EMPLOYEES ABOVE AVERAGE";H
230 PRINT"NUMBER OF EMPLOYEES BELOW AVERAGE";L
240 PRINT"NUMBER OF EMPLOYEES AT AVERAGE";E
250 END

```


5. Suppose Mrs. Johnson now needed to know what each employee's variance from the average salary was. How could this be determined? (Find the difference between the average salary and each employee's salary.)
- . Forecast the outcome when the following lines are added to the program we have written, knowing that the average salary is \$19429.70.

```

250 RESTORE
260 READ A$, B
270 IF B= -1 THEN 300
280 PRINT "The variance of "A$"'s salary from the average is ";AV-B
290 GOTO 260
300 END

```

Answer:

<u>Employee</u>	<u>Variance</u>
SUSAN ADAMS	2979.70
HENRY DUZANT	4929.70
BARUCH JONES	1496.70
MARISSA MORRIS	-4470.30
LOUISE TUPPER	1679.70
HOM-HONG WEI	-6525.30

6. Why was the variance for Marissa Morris and Hom-Hung Wei a negative number? (because the average salary was less than their salary and the formula used was average salary minus employee's salary)
- . If the variance is the difference between the average salary and the employee's salary, should there be a negative variance? (no)
7. What is absolute value? (the distance a number is from zero)
- . Use of absolute value is a means of preventing the variance from being a negative number.
- . An ABS statement will compute the absolute value of a number.
8. How may we incorporate the ABS statement into the program?

Acceptable answer:

```

280 PRINT "The variance of "A$"'s salary from the average is ";ABS(AV-B)

```

Applications:

1. Determine the output of each program:

```

(a) 10 X = 7
    20 Y = 4
    30 Z = (X + Y) - (X - Y)
    40 Print ABS(Z)
    50 END

```

```

(b) 10 INPUT "How old are you?";Y
    20 INPUT "How old is your best friend?";B
    30 V =ABS(Y-B)
    40 PRINT"Your ages vary by " V " years"
    50 END

```

2. Using READ/DATA statements, write a program that will determine the average age of five of your friends and then will determine each person's variance from the average.

Summary:

1. How can DATA be READ more than one time within a program without reentering the same DATA?
2. How does the internal pointer handle sets of DATA values when a RESTORE statement is used?

Assignments:

1. Write a program that will determine the average number of words typed by five secretaries and will then output the names of the secretaries who typed more than the average amount. The DATA is:

<u>Secretary</u>	<u>Words per minute</u>
Mr. Kogan	38 wpm
Mrs. Miller	75 wpm
Mr. Reali	80 wpm
Mrs. Zorn	45 wpm
Ms. Wright	53 wpm

2. Write a program that will READ the following set of DATA, will output the largest number, and then, using the RESTORE statement, will output the smallest number.

Data: 5, -7, 3, 10, 0, -2, 1, -1

3. Determine what the output of the following program will be:

```

10 READ X
20 REM*TEST FOR IMAGINARY NUMBERS
30 IF X <0 THEN S$="I"ELSE S$=" "
40 Z=ABS(X)
50 Y=SQR(Z)
60 PRINT"THE SQUARE ROOT OF ";X;" IS "Y;S$
70 DATA 9, 400,-25, 500, -60, 100
80 END

```

4. Write a program that will print out stars in specific screen locations, using the TAB statement and the DATA provided below to demark specific locations. After the first printing the computer should print out stars in every second location, in the next printing every third location,

and so on, until the computer has completed this process five times, eventually printing in every fifth location.

Data: 42,120,500,321,299,609,812,1000,214,798,467

5. Mrs. Graham's class of ten students receives a report quarterly concerning their grade point average in English. Create a program that will store within a DATA statement the names of the ten students and the marks they have received during the term. The computer should then output each student's average, the class average, the names of those students whose average is above 90, and the names of all students whose average is below the class average.

Topic 6

NESTED FOR/NEXT LOOPS

Performance Objectives: Students will be able to:

- . recognize the need for a nested loop
- . define a nested FOR/NEXT loop
- . use the nested FOR/NEXT loop in a program

Vocabulary: New: nested FOR/NEXT loop

Review: FOR/NEXT, IF/THEN, READ/DATA, INPUT, enhanced INPUT, string variable, time-delay

Motivation:

- . How can we write a program that will ask the user to type in a number as a word and as a numeral, then print out the word in a column the number of times suggested by the numeral? For example, if the number chosen is three, the computer should create the following output:

```
THREE
THREE
THREE
```

- . What is the output for this program:

```
10 INPUT"TYPE YOUR FAVORITE NUMBER IN WORDS";X$
20 INPUT"TYPE THIS NUMBER AS A NUMERAL":X
30 FOR I = 1 TO X
40 PRINT X$
50 NEXT I
60 END
```

(output if favorite number is two:

```
TWO
TWO )
```

- . What two lines of the program did line 10 replace?

Answer:

```
PRINT "Type your favorite number in words"
INPUT X$
```

- . What type of input statement is illustrated by line 10? (enhanced input statement)
- . In the statement for I = 1 to X, what does X represent? (X will tell computer to perform an operation X number of times.)
- . When a FOR statement is found within a program, what other statement will be found within the program? (NEXT)

- . What occurs on line 50? (Line 50 tells the computer to go back to line 30, check that the operation has been performed, and then perform it again.)
- . How many times will the loop be executed? (X number of times)
- . How does the computer know that once line 50 is executed the control passes to line 30 and not to line 10 or line 20? (The NEXT statement tells the computer to increment X by one and then branch back to the beginning of the loop, indicated by the FOR statement.)
- . Does the computer ever reach line 60? If so, when? (The computer reaches line 60 after it has executed the loop X number of times. After executing the program X number of times, it then proceeds to the next line number after line 50, which in our example is 60 END.)
- . How quickly is the output printed on the screen? (Stress that this is done almost instantly.)
- . Suppose we wanted the computer to print its output more slowly. What is needed to create this slowdown? (a time-delay)
- . What does a time-delay do? (The computer counts to itself before continuing.)
- . Where, in this program can a time-delay be included to slow down the output of the program? (somewhere between line 30 and line 50)
- . How can this be accomplished? (by using a counter that will increment before printing)
- . Would it be possible to insert a FOR/NEXT statement inside the FOR/NEXT loop that is already there?

Aim: How can we use a FOR/NEXT loop within another FOR/NEXT loop?

Development:

1. We would like to create another FOR/NEXT loop within an existing FOR/NEXT loop. By doing this, we are nesting FOR/NEXT loops.
 - . Show that the nesting function is really building a loop within a loop.

```

30 FOR I = 1 to X
40 PRINT X$
42 FOR J = 1 TO 100
44 NEXT J
50 NEXT I

```
2. What does line 30 do? (Line 30 tells the computer that a command will be executed X times.)
3. What command will be executed X times? (lines 40 to 44)

4. What does line 42 do? (Line 42 tells the computer that a command will be executed 100 times.)
5. What command will be executed 100 times? (NEXT J)
Will anything be output while the computer is counting 100 times? (no)
6. When will the computer return to line 30? (after it has completed the loop of lines 42 and 44)
7. How does the computer know to return to line 30 and not to line 44 after completing line 50? (NEXT I is the command on line 50, which sends the computer back to line 30 to check and see that the operation has been performed X times.)
8. How can the time-delay be lengthened to slow down the printing of the numbers? (Change line 42 so that J = 1 to a number greater than 100.)
9. How can the time-delay be shortened to quicken the speed of the output? (Change line 42 so that J = 1 to a number smaller than 100.)
10. Show graphically how the loops may be traced:

```

10 INPUT"TYPE YOUR FAVORITE NUMBER IN WORDS";X$
20 INPUT" TYPE THIS NUMBER AS A NUMERAL";X
30 FOR I = 1 TO X
40 PRINT X$
42 FOR J = 1 TO 100
44 NEXT J
50 NEXT I
60 END

```

11. When I = 1, what value does J have? (J varies from 1 to 100.)
12. How many times is J incremented before the next I occurs? (J is incremented 100 times.)
13. Rewrite line 40 as follows and forecast the output:

```

40 Print I, X$

```

Output:

1	THREE
2	THREE
3	THREE

Applications:

1. Predict the outcome of the following examples:

```

(a) 10 PRINT"We will print the set of all 2-digit numbers that contain
      only the digits 1, 2, and 3"
      20 FOR I = 1 to 3
      30 FOR J = 1 to 3

```



```

40 PRINT I; J
50 X = X + 1
60 NEXT J
70 NEXT I
80 PRINT "There are " X "2-digit numbers that contain only the
  digits 1, 2, and 3"
90 END

```

Outcome:

We will print the set of all 2-digit numbers that contain only the digits 1, 2, and 3.

```

1 1
1 2
1 3
2 1
2 2
2 3
3 1
3 2
3 3

```

There are nine 2-digit numbers that contain only the digits 1, 2, and 3.

```

(b) 10 X$ = "Good"
    20 Y$ = "BYE"
    30 FOR A = 1 to 2
    40 PRINT X$
    50 FOR B = 1 to 3
    60 PRINT Y$
    70 NEXT B
    80 NEXT A
    90 END

```

Outcome:

```

Good
Bye
Bye
Bye
Good
Bye
Bye
Bye

```

(c) Modify program (b) to include a time-delay so there will be a pause before the word "Good" is printed.

Answer:

```

35 FOR C = 1 to 200
37 NEXT C

```

2. Write a program that will print the following numbers:

```
1  1  1
1  1  2
1  1  3
1  2  1
1  2  2
1  2  3
```

Acceptable answer:

```
10 X = 1
20 FOR I = 1 TO 2
30 FOR J = 1 TO 3
40 PRINT X ; I ; J
50 NEXT J
60 NEXT I
70 END
```

. How many loops were needed? (2)

. What was the inner-most loop?

```
30 FOR J = 1 TO 3
40 PRINT X; I; J
50 NEXT J
```

. How did the computer know when to stop? (After the computer checked line 20 to see if it was completed, it then proceeded to line 70 which had the command END.)

Summary:

1. What are nested loops? (Nested loops are loops that are found within other loops.)
2. How are nested loops executed? (The outer-most loop is executed first, followed by the next inner-most loop, and so on.)

Assignments:

1. Write a program that will output the first 1000 even numbers, including a time-delay to slow down the print.

Acceptable answer:

```
10 REM *This loop will multiply*
20 FOR I = 1 TO 1000
30 PRINT I*2
40 REM **This loop will waste time**
50 FOR J = 1 TO 100
60 NEXT J
```

```
70 NEXT I
80 END
```

2. Write a counting program that will ask where you want to begin, with what number you want to end, and with what increment you want to count, and will then output the numbers requested.

Acceptable answer:

```
10 INPUT "What is the first number to be counted from";A
20 INPUT "What is the last number you want to count to";B
30 INPUT "What increment would you like to count by";C
40 FOR I = A TO B STEP C
50 PRINT I
60 FOR J = 1 TO 5
70 NEXT J
80 NEXT I
90 END
```

3. Write a program that will print out a star pattern like the following:

```
*****
      :
      :
      :
      :
*****
      :
      :
      :
```

Acceptable answer:

```
10 FOR I = 1 TO 2
20 PRINT TAB(27)"*****"
30 FOR J = 1 TO 3
40 PRINT TAB(30)"::::"
50 NEXT J
60 NEXT I
70 END
```

Topic 7 SUBROUTINES

Performance Objectives: Students will be able to:

- . write programs with subroutines to avoid duplication of code
- . explain the difference between GOSUB and GOTO

Vocabulary: New: GOSUB, RETURN, STOP, subroutine
Review: GOTO

Motivation:

- . Consider the following program segment (from steps 10 to 40) which appears three times in a long program:

```
10 HOME : REM OR CLS OR SHIFT CLR/HOME
20 FOR I=1 TO 250
30 PRINT "PLEASE HAVE PATIENCE."
40 NEXT I
1060 HOME
1070 FOR I=1 TO 250
1080 PRINT "PLEASE HAVE PATIENCE."
1090 NEXT I
3240 HOME
3250 FOR I=1 TO 250
3260 PRINT "PLEASE HAVE PATIENCE."
3270 NEXT I
```

- . Can you simplify the above program so that the program segment (from steps 10 to 40) does not have to be repeated three times?

Aim: How can programs be written to avoid duplication of code?

Development:

1. Assume that the program segment from steps 10 to 40 is written at the end of the program. What BASIC statement(s) are there which allow the program to proceed out of sequence in order to branch to the program segment on three occasions?
2. The difficulty with using a GOTO statment to branch to the program segment at the end of the program is that the computer does not know to which of the three places in the program to return.
3. There is a new BASIC term, GOSUB, which not only allows a program to branch out of sequence (as does GOTO), but also keeps track of where the branching originated. The program returns to the statement immediately following the GOSUB statement when the statement RETURN is reached. The sequence of steps where GOSUB branches is called a subroutine. For example, the motivation problem might be rewritten as follows:

```

10 GOSUB 10000
1060 GOSUB 10000
3240 GOSUB 10000
9999 STOP
10000 HOME
10010 FOR I=1 TO 250
10020 PRINT "PLEASE HAVE PATIENCE."
10030 NEXT I
10040 RETURN

```

. The statements 10000 through 10040 form a subroutine. The program will return to the statement immediately following the GOSUB in either statement 10, statement 1060, or statement 3240.

4. The only way for the program to enter our subroutine is via a GOSUB statement. For example, the above program has a STOP statement before the subroutine so that the program ends and does not enter the subroutine in the normal sequence of steps. If the subroutine is entered without a GOSUB, there will be a RETURN WITHOUT GOSUB error message.

Application:

1. Write a program to enable a user to input his or her name and address at several points in a program. Use a subroutine to avoid duplication of code.

Summary:

1. How is a GOSUB used in conjunction with a RETURN?
2. What is a subroutine?
3. What is the difference between a GOSUB and a GOTO?
4. How can one avoid getting a RETURN WITHOUT GOSUB error message?

Assignments:

1. What is the output of the following program?

```

10 GOSUB 1000
20 GOSUB 2000
30 PRINT "OR NOT"
40 GOSUB 2000
50 GOSUB 1000
60 PRINT "THAT WAS NOT THE QUESTION."
70 STOP
1000 PRINT "TO"
1010 RETURN
2000 PRINT "BE"
2010 RETURN

```

2. Correct the errors in the following program:

```
10 INPUT X,Y
20 GOSUB 100
30 INPUT X,Y
40 GOSUB 100
50 INPUT X,Y
60 GOSUB 100
100 Z=X*Y
110 PRINT Z
120 END
```

3. Write a program to print a song with a repeating choral part. Use a subroutine to avoid duplication of code.

Assignments for More Advanced Students:

1. Write a program to input a 4-by-5 array. Print the original array, an array whose elements are squares of the original elements, and an array whose elements are twice the original elements. Use a subroutine to save writing the array printing routines more than once.

Topic 8 ALGORITHMS, MODULES

Performance Objectives: Students will be able to:

- . write an algorithm for a program
- . separate a program into modules, with each module serving a different purpose

Vocabulary: New: algorithm, modules

Review: subroutine, GOSUB, RETURN

Motivation:

- . What are the steps one would need to determine the monthly payments for a \$100,000 mortgage at 10% yearly interest? Assume the mortgage is to be paid out over a 10-year period.

Aim: How can we set up programs that break up into modules (sections) serving different purposes?

Development:

1. It is easier to write a program when the problem is clearly understood. The list of steps needed to manipulate the data of a problem is called the solution algorithm. An algorithm breaks a problem down into smaller and smaller subproblems. This results in greater and greater refinement of the problem.
2. Each subproblem can then be written as a separate section, that is module, of the program. Each module has a specific function. If the program does not work, it is then easier to pinpoint the error.
3. Consider the motivation problem. The solution algorithm is:
 - . Since there are 120 months in 10 years, the monthly payment of principal is
$$\frac{\text{principal}}{120}$$
 - . The new balance owed after each month is
$$\frac{\text{old balance} - \text{principal}}{120}$$
 - . The interest paid each month is 0.10 times balance.
 - . The total monthly payment is the monthly payment of principal plus the interest.
4. The modules are:

- . Finding the monthly payment
- . Finding the new balance
- . Finding the interest per month
- . Finding the total monthly payment

Application:

1. Determine an algorithm and modules for a program to print a depreciation schedule for equipment over 5, 10, and 15 years.

Summary:

1. What is an algorithm?
2. What is the reason for separating a program into modules?

Assignments:

1. Determine an algorithm and modules for a program to print the income taxes for the employees of a company.
2. Determine an algorithm and modules for a program to print the averages for each student in a graduate class and to print the names of students who have failed more than one subject.
3. Determine an algorithm and modules for a program to solve linear equations and to graph linear functions.

Assignment for More Advanced Students:

1. Write an algorithm and modules for determining if a number is prime or composite.

Topic 9

ARRAYS: SUBSCRIPTED VARIABLES

Performance Objectives: Students will be able to:

- . define the words array, element and subscript
- . explain the purpose of the DIM statement
- . use an array to store and then print out a set of numbers

Vocabulary: New: array, subscripted variable, element, DIM

Motivation:

- . How can we write a program that will input three numbers, find their sum, and then print both the individual numbers and the sum of the numbers?

Acceptable answer:

```
10 INPUT "ENTER A NUMBER"; A
20 SU = SU + A
30 INPUT "ENTER A NUMBER"; B
40 SU = SU + B
50 INPUT "ENTER A NUMBER"; C
60 SU = SU + C
70 PRINT "THE NUMBERS ADDED WERE "; A; " "; B; " "; C
80 PRINT "THE SUM OF THE 3 NUMBERS IS "; SU
```

Aim: How can we use an array to store and then print a set of numbers?

Development:

1. Refer to the above program and challenge the students to modify it so that the program will work for an indeterminate amount (ANY number) of numbers.
 - . Elicit that a program that will calculate the sum of an indeterminate amount of numbers can easily be written:

```
10 INPUT "ENTER HOW MANY NUMBERS WILL BE ADDED"; A
20 FOR I = 1 TO A
30 INPUT "ENTER A NUMBER"; B
40 SU = SU + B
50 NEXT I
60 PRINT "THE SUM OF "; A; " NUMBERS IS "; SU
```

Elicit further that while such a program can easily be written, it is not possible to print the individual numbers since each new number was assigned to the B variable (put into B) thus erasing the previous value that was stored there.

Note: Some students might suggest using individual variables (such as A, B, C, D, E) to store the numbers. This is acceptable when we

know the number of grades to be entered while we WRITE the program; reserving an infinite number of variables in the program for possible future use is impractical.

- Explain that today's lesson will deal with a method that will allow us to calculate and print the sum of an indeterminate amount of numbers while also printing the individual numbers.

2. Use the following analogy:

A school stores its supplies in bookcases. One of the bookcases is labeled stationery and another sports. Each has five shelves.

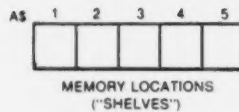
- State that paper is stored on shelf 1 of the stationery bookcase, labels on shelf 3, and pens on shelf 4; that on shelf 2 of the sports bookcase we would find a bat and on shelf 5 a glove.

STATIONERY		SPORTS	
1	PAPER	1	
2		2	BAT
3	LABELS	3	
4	PENS	4	
5		5	GLOVE

- State that we could send someone to the stationery bookcase and ask for what is on shelf 4. The person would bring pens. Similarly, we could send someone to bring what is on shelf 2 of the sports bookcase and the person would return with a bat.
 - Explain that the computer also has "bookcases" with "numbered shelves" inside it and that these are called ARRAYS.
3. Show how the names of the five students in a class could be stored in an array.
- Call the array class. We could have the computer put John on shelf 1, Mary on shelf 2, Eric on shelf 3, Jane on shelf 4, and Joan on shelf 5.

CLASS	
1	JOHN
2	MARY
3	ERIC
4	JANE
5	JOAN

- Explain that just as we could get paper from shelf 1 of the stationery bookcase so, too, can we tell the computer to print what is on shelf 4 of the array called class. The computer would print Jane. If we asked it to print what is on shelf 2 it would print Mary.
- Indicate that if we told the computer to PRINT what is on shelf 3", we would get an SN error message -- the language we are using is not understood by the computer.
- Explain that the computer organizes its memory spaces like a bookcase; the computer puts the name of a variable in one spot and then marks off the succeeding memory locations as the "shelves" of the variable.



4. List the following BASIC syntax for arrays:

- . Arrays are named in almost the same way we name regular variables, the only difference being the parentheses with the "shelf number" inside them.
- . A\$(N) is the name of an array which holds string variables.
- . A(N) is the name of a numeric array.
- . The (N) refers to the number of the "shelf" of the array.
- . Programmers refer to the "shelf" as an element of the array.
- . A(2) is pronounced "A sub two." This is why arrays are also called subscripted variables.
- . The statement A\$(2) = "A" says that there is an "A" on "shelf" 2 of the array A\$ or that element 2 of array A\$ has a value of A.

5. Before an array can be used in a program we have to tell the computer how many elements we wish to use. We do this with a DIM statement. (DIM stands for DIMension.)

For example:

```
10 DIM A$(20)
```

This tells the computer to reserve space for an array called A\$ which will have 20 elements. This statement should be the first line of the program.

6. Referring to the class list mentioned in Development 3, ask the students how we could load an array with the the names of the five class members.

- . Elicit that since all we are doing is assigning a value to a variable we can use any of the three assignment statements we learned previously, LET, INPUT or READ/DATA.
- . For this example use the LET statement and elicit the following program:

```
10 DIM A$(5)
20 LET A$(1) = "JOHN"
30 LET A$(2) = "MARY"
40 LET A$(3) = "ERIC"
50 LET A$(4) = "JANE"
60 LET A$(5) = "JOAN"
```

- . Have the students add to the above program so that it outputs the list of students to the screen.

Acceptable answer:

```
70 PRINT A$(1)
80 PRINT A$(2)
90 PRINT A$(3)
100 PRINT A$(4)
110 PRINT A$(5)
```

7. Refer to the Motivation and have the students use an array to write a program that will load any five numbers into an array, calculate the sum, and then print the sum and the individual numbers.

Acceptable answer:

```
5 CLS : REM HOME or SHIFT HOME/CLR
10 DIM A(5)
20 INPUT "ENTER A GRADE"; A(1)
30 LET SUM = SUM + A(1)
40 INPUT "ENTER A GRADE"; A(2)
50 LET SUM = SUM + A(2)
60 INPUT "ENTER A GRADE"; A(3)
70 LET SUM = SUM + A(3)
80 INPUT "ENTER A GRADE"; A(4)
90 LET SUM = SUM + A(4)
100 INPUT "ENTER A GRADE"; A(5)
110 LET SUM = SUM + A(5)
120 PRINT "THE INDIVIDUAL NUMBERS ADDED WERE "; A(1); " "; A(2); " "; A(3); "
    "; A(4); " "; A(5)
130 PRINT "THE SUM OF THE 5 NUMBERS IS "; SUM
```

- . Elicit that the computer could do more of the work if FOR/NEXT loops were used.

Acceptable answer:

```
5 CLS
10 FOR I = 1 TO 5
20 INPUT "ENTER A NUMBER"; A(I)
30 LET SU = SU + A(I)
40 NEXT I
50 CLS
60 PRINT "THE INDIVIDUAL NUMBERS ADDED WERE ";
70 FOR I = 1 TO 5
80 PRINT A(I); " ";
90 NEXT I
100 PRINT
110 PRINT "THE SUM OF THE 5 NUMBERS IS "; SU
```

Application:

1. Write a program that will input 10 grades, calculate the average, and then print out both the individual grades and the average.

Acceptable answer:

```
5 CLS
10 DIM A(10)
20 FOR I = 1 TO 10
30 INPUT "ENTER A GRADE"; A(I)
40 LET SU = SU + A(I)
50 NEXT I
60 CLS
70 PRINT "THE INDIVIDUAL GRADES WERE ";
80 FOR I = 1 TO 10
90 PRINT A(I); "    ";
100 NEXT I
110 PRINT
120 PRINT "THE AVERAGE OF THE 10 GRADES IS "; SU/10
```

Summary:

1. What is an array?
2. What are three ways we can assign a value to an array element?
3. How can we print out the value of an array element?
4. What is the difference between A\$ and A\$(I)?
5. What is the difference between a subscript and the value of a subscripted variable?

Assignments:

1. Study the following program:

```
10 LET B = 6
20 LET A(B) = 9
```

- (a) What is the value of element A(B)?
- (b) Which element has a value of 9?
- (c) Is there an element in the array whose value is 6? Which one?
- (d) How many elements were reserved at the beginning of the program?
- (e) How many elements were used?

2. What is the output of the following program?

```
10 DIM A$(15)
20 FOR I = 1 TO 3
30 READ A$(I)
40 PRINT A$(I)
50 NEXT I
60 DATA "BOB", "JANE", "JIM"
```

Where in the array would we find JANE?

3. Write a program that will input the first 10 letters of the alphabet,

store them in an array, and then print the list backward. The PRINT statement will also identify the array element each letter occupies and will look like this:

LETTER J IS FOUND IN THE FIRST ELEMENT OF THE ARRAY

4. Use READ/DATA to enter a list of six names into an array. Have the program print the list of names in the center of the screen. Each name should not be longer than eight letters.
5. Write a program which will store 10 numbers in an array and will print the 2nd, 4th, 6th, 8th, and 10th number of the array.

Assignments for More Advanced Students:

1. Write a program in which the computer will randomly choose 10 numbers between 1 and 50, will place them in an array, and then will print the list of numbers along with the number of the element they occupy.
2. Change the above program so that there will be no duplication of numbers.
3. Change the above program so that the computer will input any amount of numbers.

Topic 10

ARRAY VARIABLES

Performance Objectives: Students will be able to:

- . explain the need for reserving string space for a string array in addition to dimensioning it
- . write a program in which they will associate elements of different arrays

Vocabulary: New: default value, clear
Review: array, element, DIM

Motivation:

- . We have written programs that use arrays and that can print a list of names. We have also written programs that can find the average of a set of grades and print both the individual grades and the average. How can we associate a class list of students stored in one array with their grades which are stored in other arrays?

Aim: How can we use arrays to associate different lists of information?

Development:

1. Review the concept of an array.

- . Refer to the following line:

```
10 LET A$(2)="BOB"
```

Make the following points:

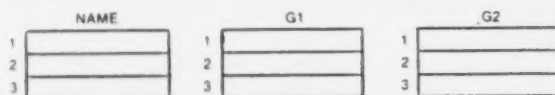
- A\$() is the name of an array which stores strings.
- The (2) in A\$(2) tells us which element of the array A\$ we are dealing with.
- "BOB" is the value assigned to the second element of array A\$.

2. Review the DIM statement.

- . Show its syntax as 10 DIM A\$(20).
- . State that DIM is short for DIMENSION.
- . Explain that it tells the computer the name of the array to be used and the number of elements that the array is to have (its size, or dimension).
- . Say that some computers will allow using an array of up to A(10) without using a DIM statement. Explain that this is called a default value.
- . State that '0' is an acceptable element number so that DIM A\$(10) provides for a total of 11 elements, A\$(0) through A\$(10).

- . Explain that if an attempt is made to refer to an element which is greater than that allowed for in the DIM statement, an error message is printed telling us that the subscript (element number) is "bad" or "out of range".
 - . State that an array is allowed to be dimensioned once in the program.
3. Refer to the bookcase analogy presented in Topic 9.
- . Remind the students that in the stationery bookcase each type of object is stored on a different shelf. The greater the number of kinds of objects to be stored, the greater the number of shelves required to hold them. Relate this to the DIM statement and the number of elements in an array. Note that the greater the amount of any one type of object, the larger the individual shelf holding it has to be.
 - . Show how the number of characters stored in string variables is analogous to the number of items stored on the individual bookcase shelf.
 - . Explain that in many computers just as the amount of array space is initially limited (to 11 elements), so too is the amount of available string variable space limited to 50 characters for all the string variable assignments used in the program.
 - . Discuss the case of the computer which has a default value of 50 string characters (the number of characters we can store in all the string variables in a program without having to ask for these spaces). Show that if a 15-element string variable array, each element of which consists of a 10-character name, is implemented on this computer the default allowable number of string space is quickly surpassed.
 - . Elicit that just as it is possible to increase the number of array elements (through the use of the DIM statement) so too should it be possible to increase the number of available string variable spaces. Some machines require that the programmer dictate exactly how much string space will be required at the beginning of each program unless a default amount (usually 100 bytes) is acceptable. Indicate that for such machines the use of the CLEAR statement is necessary.
 - . A typical statement on a TRS80 model I or III would read CLEAR 1000.
4. Introduce the following problem:
- Three students named Bob, Ted, and Alice took two tests each. How can we write a program which would store the names of the students in one array, their grades on the first exam in a second array, and their grades on the second exam in yet another array?
- . Challenge the students to think about writing their program in such a way that the program could be later expanded first to print the students' names and grades and then to calculate and print the average of their grades.

- Ask what type of arrays are needed, string or numeric. Elicit that a string array is needed for the names while the grades, which could be stored in either type, are placed in a numeric array because of the subsequent need to calculate with them.
- Ask how it is possible to associate the names with the grades if they are stored in different arrays. Elicit that if each student's name and grades are placed in the same element number of their respective arrays the information can easily be tied together.
- Reinforce this idea by putting the following diagram on the board:



- Show, by using the data below, that if we put the student's name in element 1 of the name array we could associate the name with the grades by placing the grades in element 1 of the other arrays.

Data: Bob, 90, 100; Ted, 94,96; Alice, 95, 98

- Have the students now write a program using the same data.

Acceptable answer:

```
10 FOR I=1 TO 3
20 INPUT "ENTER THE STUDENT'S NAME"; A$(I)
30 INPUT "ENTER THE FIRST GRADE"; G1(I)
40 INPUT "ENTER THE SECOND GRADE"; G2(I)
50 NEXT I
```

Application:

1. Have the students expand the above program so that it will print the following table with the relevant data filled in:

CLASS LIST			
NAME	GRADE 1	GRADE 2	AVERAGE
----	-----	-----	-----

Acceptable answer:

```
10 FOR I = 1 TO 3
20 INPUT "ENTER THE STUDENT'S NAME"; A$(I)
30 INPUT "ENTER THE FIRST GRADE"; G1(I)
40 INPUT "ENTER THE SECOND GRADE"; G2(I)
50 NEXT I
60 CLS ' OR HOME
70 PRINT "NAME"; TAB(10); "GRADE 1"; TAB(20); "GRADE 2"
80 PRINT "----"; TAB(10); "-----"; TAB(20); "-----"
90 PRINT
100 FOR I = 1 TO 3
110 PRINT A$(I); TAB(13); G1(I); TAB(23); G2(I)
120 NEXT I
```

Summary:

1. Under what circumstances do we have to dimension arrays?
2. When and why do we have to pay strict attention to clearing extra string space when using string arrays?
3. How can we link together information contained in four different arrays?

Assignments:

1. What would be the value of A\$(17) after the following program is executed?

```
10 DIM A$(16)
20 FOR I = 1 TO 20
30 INPUT A$(I)
40 NEXT I
```

2. Expand the Application problem to have it also calculate the average and print it in its own column.
3. Write a program that will use READ/DATA to input the last names, street addresses, city, state, zip code, and telephone number of five people. The data will be stored in six different string variable arrays. The program will then output the information using commas to separate the items. Each individual should have his or her own line(s). The printout will look like this:

```
JONES, 287 NINTH STREET, BROOKLYN, NEW YORK, 11229
212 - 645-1334
```

4. Write a payroll program for a company of five employees. It will use READ/DATA to input the following data:

employee's last name, hours worked, rate of pay

After all the employee data is input the program will calculate the employees' gross pay and will print the following table heading:

PAYROLL - NATIONAL NONSENSE COMPANY
Week of April 1

NAME	HOURS WORKED	RATE OF PAY	GROSS PAY
------	--------------	-------------	-----------

It will then fill in the table with the relevant data.

5. Write a program that will input the names of the preceding seven days, the weather type (sunny, cloudy, rainy, etc.), and the temperature of each day. The data should be stored in separate arrays. The program

should then print out the following table heading and use the input data to fill it in.

WEATHER WEEK

DAY	WEATHER TYPE	TEMPERATURE
-----	--------------	-------------

Assignment for More Advanced Students:

1. Write a payroll program that will be used to calculate the salary of the employees of the New York Pothole Corporation. Each employee gets paid time-and-a-half for all hours worked above the normal 40-hour week. If the employee is a union member then dues of \$10 are deducted from the pay.

Tax is deducted as follows:

Weekly gross	Tax
-----	---
\$200 - \$299	10%
\$300 - \$399	15%
\$400 - \$499	20%
\$500 and above	35%

The employee can contribute up to 15% of the gross salary to a pension plan. The pension deduction is deducted from the gross before the taxes are calculated.

Write a program which will calculate the net salary of an employee. Part of the task of writing this program is for you to supply the data for it. You must include data to test six different combinations. For example, there should be one employee who worked 40 hours, was a union member, contributed 15% of his salary to a pension, and earned \$475 per 40-hour week. Make sure that you include someone who worked less than 40 hours and someone who worked more than 40 hours.

The program will print out the following:

EMPLOYEE NAME: JONES, SAM
HOURS WORKED : 40
WEEKLY RATE : \$375
GROSS PAY : \$375
OVERTIME PAY : ----
UNION DUES : \$10
PENSION : \$15%
TAXES PAID :
NET PAY :
HOURS WORKED : 40

57

Topic 11

PROGRAMS WITH SUBROUTINES

Performance Objectives: Students will be able to:

- . write programs with subroutines as modules

Vocabulary: Review: modules, subroutines, GOSUB, RETURN

Motivation:

- . Consider the following:

```
10 REM MODULE ONE
20 INPUT X(1),X(2),X(3),X(4),X(5)
100 REM MODULE TWO
110 DATA 135,389,983,276,903
120 FOR I=1 TO 5
130 READ X(I)
140 NEXT I
200 REM MODULE THREE
210 X(1)=852:X(2)=481:X(3)=87:X(4)=826:X(5)=61
300 REM MODULE FOUR
310 PRINT (X(1)+X(2)+X(3)+X(4)+X(5))/5
```

Rewrite the program so that modules one, two, and three utilize module four (Write module four as a subroutine.)

Aim: How can we write programs that utilize subroutines as modules?

Development:

1. Consider a program to determine the monthly payment for a mortgage of \$100,000 at 10% over a 10-year period. (The algorithm and modules were developed in the previous lesson.)
2. The program is:

```
5 REM THE VARIABLE PR STANDS FOR PRINCIPAL WHILE YE STANDS FOR YEARS AND
  RA FOR RATE.
10 INPUT PR, RA, YE
20 GOSUB 1000
25 FOR I=1 TO 120
30 GOSUB 2000
40 GOSUB 3000
50 NEXT I
60 STOP
1000 REM MONTHLY PRINCIPAL PAYMENT
1010 M=PR/(12*YE)
1020 RETURN
2000 REM NEW BALANCE OWED EACH MONTH
2010 NB=PR - M
2020 RETURN
3000 REM MONTHLY INTEREST PAYMENT
3010 MP=RA*NB
3020 RETURN
```

Application:

1. Write a program to print a depreciation schedule for equipment over 5, 10, and 15 years.

Summary:

1. What are the advantages of writing a module as a subroutine?

Assignments:

1. Using a subroutine, write a program to print the taxes for the employees of a company.
2. Using a subroutine, write a program to print the averages for each student in a graduating class and the names of students who failed more than one subject.
3. Using a subroutine, write a program to solve linear equations and to print tables of values that could be used to graph linear functions.

Assignments for More Advanced Students:

1. Using a subroutine, write a program to determine if a number is prime or composite.
2. Using a subroutine, write a program to print the largest factor for a series of numbers. Use the following algorithm for any given number x :
 - (a) Find the nearest integer $\leq x/2$.
 - (b) If $x/(x/2) = \text{INT}(x/(x/2))$, then $x/2$ is the largest factor.
 - (c) (1) If (b) is not true, then subtract 1 from $x/2$.
(2) Then, if $x/((x/2)-1) = \text{INT}(x/((x/2)-1))$, $(x/2)-1$ is the largest factor.
 - (d) If (c)(2) is not true, then subtract 1 from the divisor and repeat (c)(2).

Topic 12

MORE COMPLICATED SUBROUTINES

Performance Objectives: Students will be able to:

- . develop a solution algorithm for more complicated problems
- . separate these problems into modules
- . write programs for these problems

Vocabulary: Review: algorithm, module, GOSUB, RETURN, subroutine

Motivation:

- . Write a program to print a tic-tac-toe board.
- . How would you store the moves for tic-tac-toe?
- . How would you check for a win or a draw?

Aim: How can we set up the algorithm, modules, and subroutines for a more complicated program, such as tic-tac-toe?

Development:

1. We will write a program to set up a tic-tac-toe board and to determine the status of the game (win, lose, or draw) for two external players.
2. The algorithm is:
 - (a) Use an array (T(9)) for the board.
 - (b) Check each move to determine if it is legal (within the range and not occupied).
 - (c) Check the rows, columns, and diagonals to determine if there is a win, that is, if the same letter fills any row, column, or diagonal.
 - (d) Check to determine if there is a draw (the board is filled up).
3. The modules are:
 - (a) Print the array after each move.
 - (b) Determine if moves are legal.
 - (c) Determine if there is a win or draw.
4. The program is:

```
10 DIM T(9)
15 REM ASSIGN POSITIONS ON BOARD
20 FOR I = 1 TO 9
30 T(I)=I
40 NEXT I
50 FOR MOVE = 1 TO 9
60 GOSUB 1000
70 GOSUB 2000
80 GOSUB 3000
```

```

90 GOSUB 4000
100 NEXT MOVE
110 END
1000 REM PRINT THE BOARD
1010 FOR I = 1 TO 9
1020 IF T(I)>0 THEN PRINT T(I),
1030 IF T(I)=-1 THEN PRINT "X",
1040 IF T(I)=-2 THEN PRINT "O",
1050 IF I=3 OR I=6 THEN PRINT
1060 NEXT I
1070 RETURN
2000 REM ENTER MOVES: -1 INDICATES X'S MOVE AND -2 INDICATES O'S MOVE
2010 IF INT(MOVE/2)=MOVE/2 THEN TURN=-1:PRINT"IT IS X'S MOVE.": GOTO 2030
2020 TURN=-2:PRINT"IT IS O'S MOVE."
2030 PRINT"ENTER ANY NUMBER FOR YOUR MOVE (1 TO 9) THAT STILL APPEARS ON
THE BOARD."
2040 INPUT NUMBER
2050 RETURN
3000 REM DETERMINE IF MOVE IS LEGAL
3010 IF NUMBER<1 OR NUMBER>9 OR NUMBER<>INT(NUMBER) OR T(NUMBER)<0 THEN
PRINT "THIS MOVE IS ILLEGAL.":END
3020 T(NUMBER)=TURN
3030 RETURN
4000 REM CHECK FOR WIN - ROWS, COLUMNS, DIAGONALS
4010 FOR RO = 1 TO 7 STEP 3
4020 IF T(RO)=-1 AND T(RO)= T(RO+1)=T(RO+2) THEN PRINT "X WINS.":END
4030 IF T(RO)=-2 AND T(RO)= T(RO+1)=T(RO+2) THEN PRINT "O WINS.":END
4040 NEXT RO
4050 FOR CO = 1 TO 3
4060 IF T(CO)=-1 AND T(CO)= T(CO+3) AND T(CO+3)=T(CO+6) THEN PRINT "X
WINS.":END
4070 IF T(CO)=-2 AND T(CO)= T(CO+3) AND T(CO+3)=T(CO+6) THEN PRINT "O
WINS.":END
4080 NEXT CO
4090 IF T(1)=-1 AND T(1)=T(5)= T(9) THEN PRINT "X WINS.": END
4100 IF T(3)=-1 AND T(3)=T(5)= T(7) THEN PRINT "X WINS.": END
4110 IF T(1)=-2 AND T(1)=T(5)= T(9) THEN PRINT "O WINS.": END
4120 IF T(3)=-2 AND T(3)=T(5)= T(7) THEN PRINT "O WINS.": END
4130 RETURN

```

Application:

1. Write a program with subroutines to enter 30 random integers in a 5-by-6 array. Print the array, the largest number in each row and column, and the largest number in the whole array.

Summary:

1. What is an algorithm for the program to set up a tic-tac-toe board and to determine the status of the game (win, lose, or draw) for two external players?
2. What are the modules for the tic-tac-toe program?

Assignments:

1. Write a program to print an array and its transpose (an array obtained by switching the rows and the columns).
2. Write a program to print the greatest common factor of two numbers.
3. Write a program to print the characters in a string in alphabetical order.
4. Write a program to calculate the number of ingredients needed for different items in a recipe.

Assignment for More Advanced Students:

1. Write a program to have the computer play tic-tac-toe against a player. The computer should always win or draw.

Topic 13

INTRODUCTION TO MID\$, RIGHTS\$, LEFT\$ STRING MANIPULATION FUNCTIONS

Performance Objectives: Students will be able to:

- . use the MID\$, RIGHTS\$, and LEFT\$ functions to determine substrings of given strings
- . use the MID\$, RIGHTS\$, and LEFT\$ functions to modify portions of a given string
- . write a program that will manipulate portions of a given string

Vocabulary: New: MID\$, RIGHTS\$, LEFT\$, substring, manipulation

Review: concatenation, strings, LEN, FOR...NEXT, suppressed line feed

Motivation:

- . How can we change names in the form of PUBLIC, JOHN Q. to JOHN Q. PUBLIC?
- . Why might we want to do this? (Records are often kept alphabetically by last name, but correspondence is traditionally addressed with first name first.)
- . How do we recognize the end of the last name? (comma)
- . What step-by-step procedure must a person follow to change PUBLIC, JOHN Q. to JOHN Q. PUBLIC?
 - Look at each space until you find the comma.
 - Write everything to the right of the comma first.
 - Leave a space.
 - Write everything to the left of the comma.
 - Omit the comma.

Aim: How do we program a computer to manipulate portions of a string?

Development:

1. How can we tell a computer to look at only part of a string?

- . Introduce the MID\$, RIGHTS\$, and LEFT\$ functions:

MID\$ (Name of string variable, starting position, how many)

RIGHT\$ (Name of string variable, how many from right)

LEFT\$ (Name of string variable, how many from left)

Example:

10 A\$ = "Do your homework daily."

Z\$ = MID\$(A\$,3,4) (Z\$ = " you")

Q\$ = MID\$(A\$,9,4) (Q\$ = "home")

R\$ = MID\$(A\$,4,9) (R\$ = "your home")

G\$ = RIGHT\$(A\$,6) (G\$ = "daily.")

H\$ = LEFT\$(A\$,6) (H\$ = "Do you")

2. Some computers allow MID\$, RIGHT\$, and LEFT\$ to appear only on the right side of an equality while other machines allow them also to appear on the left side. Students should be able to simulate a computer in this area, because some machines have equivalent substring functions which can appear on both sides of the equal sign. When these functions appear on the left they cause the replacement of parts of an existing string. For example:

```
10 B$ = "COMPUTERS CAN BE HELPFUL TO ALL WOMEN."  
20 LEFT$(B$,9) = "VACATIONS"  
21 RIGHT$(B$,8) = "MEN. "  
22 MID$(B$,18,4) = "REST"  
23 PRINT B$  
    (Vacations can be restful to men.)
```

Note: Remember that this may not work on your school machines.

3. How can we program a computer to print one letter at a time, each new letter underneath the preceding one? (W\$ = "DOWN")

Acceptable answer:

```
10 W$ = "DOWN"  
20 PRINT MID$(W$,1,1)      ...also PRINT LEFT$(W$,1)  
30 PRINT MID$(W$,2,1)  
40 PRINT MID$(W$,3,1)  
50 PRINT MID$(W$,4,1)      ...also PRINT RIGHT$(W$,1)
```

4. Using a FOR...NEXT loop, how can we program a computer to write out one letter of "DOWN" at a time?

Acceptable answer:

```
10 W$ = "DOWN"  
20 FOR P = 1 to 4  
30 PRINT MID$(W$,P,1)  
40 NEXT P
```

5. Suppose W\$ = "Happiness", how would line 20 in the above program have to be modified, and why?

```
20 FOR P = 1 to 9      Length of W$ is different
```

6. What function enables the computer to count the number of characters in a string?

LEN (Name of string variable) for LENGTH of a string

7. Write a module that will print out the letters of a string, one underneath the other, no matter what the length of the string.

Acceptable answer:

```
100 L = LEN(S$)
110 FOR P = 1 to L
120 PRINT MID$(S$,P,1)   or   120 M$= MID$(S$,P,1)
130 NEXT P                125 PRINT M$ )
```

8. How could the above routine be modified so that the letters will all appear on the same line but with a space after every letter.

```
120 PRINT MID$(S$,P,1)   or 125 PRINT M$;" "
```

Applications:

1. Type the statements listed below on your computer. What might account for your results?

```
K$ = "Honesty is the best policy"
PRINT MID$(K$,3,4)
PRINT LEFT$(K$,2)
PRINT RIGHT$(K$,3)
MID$(K$,16,4) = "hard" : PRINT K$
```

2. Write a program that will print the word "upside-down" one letter at a time, each one underneath the preceding one, starting with the last letter and working toward the first letter.

Summary:

1. Name three functions that allow us to manipulate strings.
2. Explain how each of the functions enables us to partition strings.

Assignments:

1. What will be the result of creating a program with these parts:

```
D$ = "KEEP COMPUTER DISKS AWAY FROM MAGNETS."
PRINT MID$(D$,9,3) + MID$(D$,5,1)+ MID$(D$,15,10)
RIGHT$(D$,8)="MAGGOTS!"
PRINT D$
```

2. Write a program that will PRINT a person's INPUT name one letter at a time with one space after each letter. Utilize a time-delay routine to emphasize the sequential nature of the printing.
3. Write a program that will take the word "TRIANGLE" and produce:

```
T
TR
TRI
TRIA
TRIAN
TRIANG
TRIANGL
TRIANGLE
```

Assignments for More Advanced Students:

1. Write a program that will take the word "TRIANGLES" and produce:

```
T
TRI
TRIAN
TRIANGL
TRIANGLES
```

2. Research and use the VAL function and a string manipulation function to write a subroutine that will avoid an error message if a user prefixes a "\$" to an INPUT reply to a question.

Topic 14

USING MID\$, RIGHTS, LEFT\$ TO SEARCH FOR A CHARACTER IN A STRING AND THEN TO REVERSE AND OTHERWISE INTERCHANGE A STRING'S PARTS

Performance Objectives: Students will be able to:

- . use the MID\$ function to write programs to search for a character or group of characters
- . use MID\$, RIGHT\$, and LEFT\$ to extract portions of a given string
- . write programs using counters to evaluate strings (how many plural words, how many of each vowel, how many one-letter words, how many two-letter words, etc.)

Vocabulary: Review: MID\$, RIGHT\$, LEFT\$, substring, manipulate, concatenate

Motivation: (continued from the previous lesson)

- . How can we change a name in the form of PUBLIC, JOHN. Q. to JOHN Q. PUBLIC?
- . Review why we might want to do this and the step-by-step procedures that a person uses to change PUBLIC, JOHN Q. to JOHN Q. PUBLIC.
- . If N\$ = "PUBLIC, JOHN Q.", write a PRINT statement using MID\$, RIGHT\$, or LEFT\$ that causes the computer to output "JOHN Q. PUBLIC".

```
20 PRINT RIGHT$(N$,7)+" "+LEFT$(N$,6)
```

- . What do we still have to learn in order to complete the solution to our problem? (Find the comma.)

Aim: How to we program a computer to search for a given character or group of characters in a given string?

Development:

1. Recall the procedure from the previous lesson of printing one letter of a string at a time.
2. Write a program that will check each character of a string for a "," and, if a comma is found, will PRINT the position in which it was found. (Use N\$ = "PUBLIC, JOHN Q.")

Acceptable answer:

```
10 N$= "PUBLIC, JOHN Q."  
20 L = LEN(N$)  
30 FOR P = 1 TO L  
40 M$= MID$(N$,P,1)  
50 IF M$ = "," THEN 100  
60 NEXT P  
100 PRINT "THERE IS A COMMA IN POSITION " P  
999 END
```

3. How many letters will there be in the last name? (P-1)
4. How many letters will there be in given name(s)? (L-(P+1) There is a space after the comma.)
5. Modify the program above so that it will print "JOHN Q. PUBLIC".


```
(100 PRINT RIGHT$(N$, (L-(P+1))) + " " + LEFT$(N$, P-1))
```
6. Suppose we wanted to change "JOHN Q. PUBLIC" to "PUBLIC, JOHN Q." What changes would we have to make in our search loop?

```
20 FOR P = L TO 1 STEP -1   The last blank is one position to the left
                           of the last name.
```

7. Write a program that will count how many letter A's there are in a given string.

```
same as before except:
50 IF M$ = "a" OR M$ = "A" THEN C=C+1
100 PRINT "There are "C" letter A's "
```

Summary:

1. What routine will enable a computer to search a string for a given character?
2. What routine will enable a computer to search a string for a set of characters with length N?

Assignments:

1. Write a program that will accept any INPUT string with last name first and rewrite it with first name first.
2. Write a program that will READ in "PASCAL", "ISOSCELES", "MISSISSIPPI" and print each word with the "S"'s removed.
3. Write a program that will read in 10 strings, look for the substring "EXP:09/85" and, when it is found, PRINT that particular string.
4. Write a program to count how many occurrences of each vowel is found within a given string.

Assignment for More Advanced Students:

1. Write a program to count how many one-letter words, two-letter words, three-letter words, etc. are found in a given string.

Topic 15

WHEN TO USE A SUBROUTINE

Performance Objectives: Students will be able to:

- . determine when to use a GOTO and when to use a GOSUB
- . improve the speed of their programs using subroutines

Vocabulary: Review: GOSUB, RETURN, subroutine, STOP, GOTO

Motivation:

- . Write a program to give students a 20-question multiple-choice spelling test. Each question will offer four possible choices. The student will receive a score at the end of the test. Half the students should write the program using GOTO statements and the other half should use GOSUB statements.

Aim: How do we know when to use a GOTO and when to use a GOSUB statement?

Development:

1. Advantages of the GOTO statement are: (1) it is simpler to use than a GOSUB statement, and (2) there is less chance of an error, such as entering a subroutine without a GOSUB.
2. Advantages of the GOSUB statement are: (1) it encourages modular programming -- that is, separating the program into separate sections; (2) it discourages the spaghetti-like structure that often results from jumping around a program with GOTO statements; and (3) it allows the program to branch back to the next statement after the GOSUB.
3. When should one use the GOTO and when should one use the GOSUB statement? (In general, use the GOTO for simpler programs. You may avoid error messages. Use the GOSUB for more complicated programs, when you have duplication of code and many different modules.)
4. To improve the speed of your programs, consider writing the subroutines at the beginning of your programs. Whenever the program encounters a GOSUB statement, the program searches for the subroutine starting at the smallest line number. If the subroutines are at the beginning of the program, they will be located sooner. Logically, however, it usually is clearer if the subroutines are at the end of the program. You must balance speed considerations with considerations of clarity.

Applications:

1. Write a program with GOTO statements and another program with GOSUB statements to input people's names with the first name first and to print the names with the last name first.
2. Write a program with GOTO statements and another program with GOSUB statements to input a student's name and marks. The program should print the student's average and whether the student receives an honor

certificate (bronze for an average of 85 to 90, silver for an average of 90 to 95, and gold for an average of 95 to 100).

Assignment:

1. Write a program with GOTO statements and then a program with GOSUB statements to:
 - (a) Print the largest number in an array
 - (b) Input a string and then print it in reverse order
 - (c) Input an array, print it, and then print the sum of the elements in each column, product of the elements in each row, and product of the elements in each column
 - (d) Add or subtract the elements of two arrays

Assignment for More Advanced Students:

1. Write a program to multiply two arrays together. It should multiply the first element of the first array by the first element of the second one, the second element of the first array by the second element of the second one, etc. The results should be stored in a third array and then printed out in numerical order.

Topic 16

SPARING USE OF VARIABLES

Performance Objectives: Students will be able to:

- . define global and local variable
- . write programs utilizing a minimum number of variables
- . state why sparing use of variables is preferred in programming

Vocabulary: New: global variables, local variables
Review: subroutines, INT function, GOSUB

Motivation:

- . Which of these programs would use less memory in a machine?

10 FOR X = 1 TO 10	10 FOR X = 1 TO 10
20 READ A(X)	20 READ A(X)
30 NEXT X	30 NEXT X
40 FOR Y = 1 TO 10	40 FOR X = 1 TO 10
50 PRINT A(Y)	50 PRINT A(X)
60 NEXT Y	60 NEXT X
70 END	70 END

- . Have the students simulate a computer by showing the memory contents and output for each of the two programs. The output should be the same, but the second program has one fewer variable and therefore requires less memory.
- . Elicit reasons why it might be beneficial to use less memory. (Amount of memory is finite in our machines so using less memory means we can do more with a given machine. Commercially, memory means money.)
- . Elicit the concept of programming elegance. (If two programs can perform the same chores, the simpler program is the better one.)

Aim: How can programs utilize memory more efficiently?

Development:

1. Define global variable and local variable. (Global variables are the main variables used throughout a program. Local variables are variables found only in subroutines.)
2. Elicit examples of routines that have variables that serve as counters. (for example, input routines, printing routines, summations, line drawing routines)
3. Warn students of the common error of using the same variable as both a local variable and a global variable, such as in a subroutine and the body of the program.

Applications:

1. Rewrite this program with a reduced number of variables:

```
10 GOSUB 1000
20 GOSUB 2000
30 PRINT
40 PRINT TAB(10) "Table of Squares"
50 PRINT
60 GOSUB 2000
70 PRINT
80 PRINT TAB(10) "NUMBER", "SQUARE"
90 PRINT TAB(10) "-----", "-----"
100 FOR X = 1 TO 30
110 PRINT TAB(12);X,"    ";X*X
120 IF INT(X/4)= X/4 THEN GOSUB 2000
130 NEXT X
140 PRINT
150 GOSUB 2000
160 GOSUB 1000
999 GOTO 9999
1000 FOR Q = 1 TO 80
1010 PRINT "*";
1020 NEXT Q
1030 PRINT
1040 RETURN
2000 FOR R = 1 TO 80
2010 PRINT "-";
2020 NEXT R
2030 PRINT
2040 RETURN
9999 END
(only the R and Q can be interchanged)
```

2. Write a program that will input the heights of every student in the class and calculate their average height. (If sorting has been taught already, they could also calculate the median, mode, and range.)

Acceptable answer:

```
10 SUM = 0
20 FOR X = 1 TO N: REM N = the number of students in the room
30 INPUT HT(X)
40 SUM = SUM + HT(X)
50 NEXT X
60 AV = SUM/N
70 GOSUB 2000
80 GOSUB 1000
90 PRINT
100 PRINT TAB(10);"STUDENT HEIGHTS in OUR CLASS"
120 PRINT TAB(10);"-----"
130 PRINT
140 GOSUB 1000
150 PRINT
160 PRINT TAB(15);"Student Number","Height "
170 PRINT TAB(15);"-----"
```

```

180 FOR X = 1 TO N
190 PRINT TAB(22);X," ";HT(X)
200 IF INT(X/3) =X/3 THEN GOSUB 2000
210 NEXT X
220 GOSUB 1000
225 PRINT
230 PRINT TAB(10);"The average height is ";AV
240 PRINT
250 GOSUB 2000
      etc.

```

Summary:

1. What is the difference between a global variable and a local variable?
2. Why should variables be used sparingly in programs?

Assignments:

1. Write a program that will read in the name, the class and five grades for each student in the class and will then print the students' names and averages in chart form. The program should use as few variables as possible. For extra credit, the program should also print out the class average.
2. Write a program that will read in the names of 20 countries and their capital cities and ask the user to match them. Use as few variables as possible.
3. Write a program to show how an INPUT principal invested at an INPUT annual rate will grow over a 20-year period. Have the output printed in chart form. For extra credit, have the user INPUT a response to a question that asks whether interest is compounded quarterly or daily rather than annually and use the rule:

New principal = Principal + Rate/(times a year)* Principal.

4. Manhattan Island was purchased for \$24 in trinkets in 1626. If the Indians had invested their money at 7% interest, compounded annually, how much would their investment be worth today? Print the record of growth in chart form, showing the growth of money at 50-year intervals and also showing this year's value as the last entry.

Topic 17

WHAT MAKES A PROGRAM USER-FRIENDLY OR PROGRAMMER-FRIENDLY?

Performance Objectives: Students will be able to:

- . write programs that contain easy-to-understand PRINT statements which enable the user to use the programs efficiently
- . write programs that contain enough REM statements to enable another programmer to modify them easily

Vocabulary: New: Update

Review: INPUT, INKEY\$, REM, PRINT, ON __GOTO, ON ERROR GOTO ____, documentation, debug, modular programming, imbedded subroutines

Motivation:

- . Which of these two programs would you rather be assigned to change or update? (You are assigned to change the percentage of copper to 7%.)

Program 1:

```
10 INPUT B,C
20 PRINT (.04*B + .05*C)/(B+C)
30 END
```

Program 2:

```
10 REM This program will calculate the % of
20 REM copper and tin in an alloy
30 PRINT "Please type in the number of tons of "
40 PRINT " 4% copper and 5% tin metals."
50 INPUT B,C
60 LET P = (.04*B + .05*C)/(B+C)
70 PRINT "The % of Cu + Tn = ";P
80 END
```

- . Elicit from the class the reasons why program 2 is better. (You can learn what the variables represent and what the program does by a quick look at the code.)
- . Explain that programs often remain with a company longer than programmers do. As conditions change, programs have to be updated. An employer doesn't want a new programmer to have to waste time trying to figure out what a program does. Time means money. It also might become necessary to modify one of your own programs after a long time, and so programs should be as readable as possible.
- . You might explain that some workers try to safeguard their jobs by making their programs difficult to read. Then when they try to fix their own programs they have to "relearn" what they did from the beginning.

Aim: How can we write programs that are easy for other people to understand and modify?

Development:

1. What is the function of the REMs?
2. What parts of a program would should be annotated with REM statements? (variables, routines, subroutines, modules)
3. If possible, pass around samples of programs used in industry or college, where nearly every line has a REMark.

Application:

1. Modify the last few programs assigned to the class.

Summary:

1. What is the purpose of a REM statement?
2. Why should complicated programs contain REMs?
3. Why might no REMs be included in a program?

Assignments:

1. Write a routine that will "paint" a portion of the screen with a character, such as the letter A. Include enough REMs to make it easy for another person to easily modify the program to paint another portion of the screen with the same character.
2. Update previously assigned programs so that they include REMs to make them more readable.
3. Determine an algorithm and write modules for a program to PRINT the taxes for 10 employees of a company. Include REMs so the program can easily be updated in the future.
4. Write a general program to print averages for each student in a class. Include REMs so the program can be easily modified to PRINT honor roll averages.

Assignment for More Advanced Students:

1. Write a computer program that will PRINT out information in chart form. Include REMs with enough information so that a future programmer could insert:
 - (a) a title
 - (b) column headings
 - (c) parameters for independent variables
 - (d) formulae for determining the value of dependent variables

Topic 18
**INPUTs, INKEY\$ (OR GET\$) THAT ALLOW EVALUATIONS
OF UPPERCASE AND LOWERCASE RESPONSES**

Performance Objectives: Students will be able to:

- . write programs that will evaluate y as well as Y responses

Vocabulary: New: ASCII

Review: INPUT, user-friendly, INKEY\$ (or GET), VAL

Motivation:

- . What are the advantages of each of the following PRINT statements:

10 PRINT "DO YOU WANT TO PLAY AGAIN?"

or

10 PRINT "DO YOU WANT TO PLAY AGAIN? TYPE 'YES' OR 'NO' AND THEN PRESS
<ENTER> OR <RETURN>."

Aim: How can programs predict various responses to a PRINT statement and accept all equivalent answers?

Development:

1. Recall that programs that are the easiest for the user to use (user-friendly) are usually the most difficult to write.
2. Have students write a program whereby the machine randomly generates a number from 1 to 5 and the user has to guess what number it is. Recall VAL and creating programs that allow for human user error.

Acceptable answer:

```
5 LET N= RND(5)   ( or   5 LET N = INT(5*RND(1)+1) )
10 PRINT "I AM THINKING OF A NUMBER FROM 1 TO 5"
20 PRINT "GUESS WHAT IT IS."
30 LET Z$= INKEY$ (or GET Z$)
40 IF Z$ = "" THEN 30
50 LET Z = VAL(Z$)
60 IF Z < 1 OR Z > 5 THEN 10
70 IF Z = N THEN PRINT "YES, THAT'S RIGHT.":GOTO 200
80 PRINT "NO," Z " IS NOT THE NUMBER I WAS THINKING OF."
90 PRINT "WANT TO GUESS AGAIN?"
100 LET YN$ = INKEY$      (or GET YN$)
110 IF YN$ = "" THEN 100
120 IF YN$ = "Y" OR "y" THEN PRINT "O.K.": :GOTO 20
130 IF YN$ = "N" OR "n" THEN 999
140 PRINT "TYPE IN EITHER 'Y' OR 'N'": GOTO 100
    . . .
200          etc.
```

2. Rewrite lines 100 to 140 using an INPUT statement.
3. Introduce ASCII (pronounced as 'key'). ASCII means American Standard Code for Information Interchange (see appendix). The ASC function returns the ASCII value of the first character of the string.
4. Suppose we were creating a game that had to deal with the Wild West in American history and we were using colloquial speech to add to the color of the program. Write a module that would use the ASC function to accept such answers as "Yup", "Yeah", and "Yes-sir-ee", all as "Yes" equivalents.

```
(1000 LET A = ASC(A$)
1010 IF A = 89 or A = 121 THEN ...)
```

Applications:

1. Write a program that would allow the user to scribble on the screen, pressing either U or u for making the line go up, D or d for down, R or r for right, and L or l for left.
2. Write a module that will allow the user to INPUT A or a as a reply meaning "GO AGAIN", E or e for "END", B or b for "GO TO THE BEGINNING AND GET INSTRUCTIONS."

Summary:

1. Name two ways we can modify a program to allow for uppercase or lowercase responses from the user.

Assignments:

1. Write a program that asks, "Were vacuum tubes or transistors used in first-generation computers?" Since it is so easy to misspell the words "vacuum" and "transistor," use the ASC function to accept incorrect spellings. Allow for capital as well as lowercase responses.
2. Write a module that could be used for "True or False" tests.
3. Write a module that could be used for filling in "always", "sometimes", and "never" blanks.
4. Look up the ASCII number for the carriage return (the <ENTER> or <RETURN> key) and write a routine to test for it. You need that code to exit INKEY\$ (or GET) loops that you have created to examine INPUT responses one character at a time. Modify your previous program modules so that they are now complete.
5. Write a program that will simulate the rolling of a pair of dice and that will print the outcome. Have the next roll proceed with the user's press of r or R. For extra credit, build in a counting device that will print the total number of rolls and the number of times each sum from 2 to 12 appeared on the dice when s or S is pressed during a given RUN of the program.

Assignments for More Advanced Students:

1. Write a program that asks a user to think of a number that the machine has to guess. The user should respond S or s for smaller, L or l for larger. Develop an algorithm to allow the computer to arrive at the answer with a minimum number of guesses.
2. Write a program that will take a user's response, will use INKEY\$ (or GET) to ignore everything except the digits 0 through 9, and will build a new string. You will need to use the ASCII for <ENTER> or <RETURN> in order to determine when the user is finished.

Topic 19

USE OF LEFT\$ TO EVALUATE PARTIAL RESPONSES

Performance Objective: Students will be able to:

- . write programs using the LEFT\$ function to evaluate partial responses

Vocabulary: Review: LEFT\$, MID\$, INPUT

Motivation:

- . In the Assignment that followed Topic 18, you wrote a program using the ASC function to help evaluate the response to the question, "Were vacuum tubes or transistors used in first-generation computers?" Suppose an ASCII table were not readily available. How could we rewrite our decision statements using the string manipulation functions? (MID\$(A\$,1,X) or LEFT\$(A\$,X) (100 IF LEFT\$(A\$,1)="V" or LEFT\$(A\$,1)="v" THEN PRINT "Correct!")

Aim: How can we use string manipulation functions to evaluate portions of user responses and minimize error messages caused by human error.

Development:

1. Write a routine using the LEFT\$ function instead of ASCII for testing true or false responses.

Acceptable answer:

```
100 IF LEFT$(A$,1)="T" OR LEFT$(A$,1)="t" THEN PRINT "CORRECT!" :R=R+1
:REM if true is correct choice
110 IF LEFT$(A$,1)="F" OR LEFT$(A$,1)="f" THEN PRINT
    "SORRY"
120 PRINT "Type in whether you believe the statement is TRUE or FALSE":
    GOTO 90
```

2. Write a program using LEFT\$(A\$,2) to evaluate the answer to "Which is the capital of Ohio? Cincinnati, Cleveland, or Columbus?" Why wouldn't ASC function help us with this answer? (All begin with C. ... By the way, Columbus is the correct answer.)
3. Write a routine that might follow this PRINT statement: "Do you want to Begin again or Stop?"
4. Write sample questions for a general knowledge test where the questions are similar to the Motivation problem: The answers are contained in the problem and start with different letters. Elicit questions from the class in different subject areas and two or more possible answers for each. If the correct answer begins with the same letter as an incorrect choice, make sure that the LEFT\$ function is modified for that question.

Summary:

1. How can LEFT\$ be used to evaluate parts of a user's response to an INPUT statement?

Assignments:

1. Write a routine using LEFT\$ that could be used for evaluating yes or no responses.
2. Write a routine using LEFT\$, that could be used for evaluating always, sometimes, and never responses.
3. Write a program that asks what state contains Rocky Mountain National Park. Since many people misspell Colorado as Colorodo or Colorada or Colarado, be prepared to accept an incorrect spelling if the idea is correct, but have a routine to tell users they've spelled it wrong, if they have.
4. Michael Jackson won a record number of Grammy awards for his albums. Stonewall Jackson was a Southern general. Andrew Jackson was President of the United States. Henry M. "Scoop" Jackson was a Senator from the State of Washington. Eddie Jackson was in vaudeville as a dancer with the team of Clayton, Jackson, and Durante. Jesse Jackson is a civil rights leader. Write a question that involves one of them and use LEFT\$ to help evaluate the response.

Assignment for More Advanced Students:

1. Fill-in questions are always difficult. Some teachers give you half credit for the right idea but the wrong spelling. Unless a computer program is especially designed to provide for it, you will get no partial credit from a computer. The brightest star (not planet) in our sky is Sirius, but that's difficult to spell. Write a fill-in test question about the brightest star and help the poorer students as much as you can. By the way, Sirius is also called the Dog Star. You might want to take that into consideration in your module.

Topic 20

PROGRAMS THAT ALLOW MULTIPLE SPELLINGS OF USER RESPONSES

Performance Objective: Pupils will be able to:

- . write programs that will accept multiple responses as correct entries

Vocabulary: Review: homonyms, homophone, OR

Motivation:

- . What would be a proper program line to follow these lines of code:

```
110 PRINT "Punched cards have been replaced as the most popular auxiliary  
storage device by floppy and hard-----"  
120 INPUT A$
```

Acceptable answer:

```
130 IF A$ = "disks" OR A$ = "discs" THEN PRINT "Correct"
```

Aim: How can programs be written that allow for multiple spellings of user responses?

Development:

1. Review the logical connector OR and its use:

```
100 IF A=B THEN PRINT "HI"  
110 IF A=C THEN PRINT "HI"  
becomes  
100 IF A=B OR A=C THEN PRINT "HI"
```

2. Elicit from the class other words that have two or more acceptable spellings (such as programing and programming, honor and honour and all those words that we spell with "or" and the British spell with "our," Xmas and Christmas, adviser and advisor).
3. Create a program in which the answers would be some of the words elicited in (2) above.
4. Elicit from the class homonyms or homophones (such as all ways and always, do-due-dew, two-to-too, blue-blew, peace-piece). Create a program that requires that the user fill in the correct word based on a definition or similar statement (for example, "What is the only even prime number?"). There should be computer responses for wrong homonyms (such as "No, 'too' means 'also'" and "No, 'to' means 'toward'"). If the INPUT response has no relation to the homonyms, then the correct answer might be given.
5. Elicit from the class some commonly misspelled words or select some from a "hundred spelling demons" list. Write sample test questions that might

generate those words as responses. Give credit for the concept, but alert the user to incorrect spellings when they occur (as well as the usual incorrect entries). Examples of spelling demons are flexible, capable (any -ible or -able suffix), separate, inconvenience, athletic, embarrass.

6. There is a way of evaluating responses by considering them equivalent if 80% of the letters in the INPUT response are in the same order as the expected answer.

Summary:

1. Name three ways programs can allow for multiple spelling of user responses.

Assignments:

1. Write a program with five questions that have one element of a set of homonyms as the correct response. (For example, "Martin Luther King, Jr. won the Nobel Prize for -----." If the response is "piece" instead of "peace," the user should get credit but be made aware of the spelling error.)
2. Write a program with five questions where the responses will be commonly misspelled words. (For example, "In order to produce meringue, it is necessary to ----- the egg whites from the yolks." "Separate" is correct, not "seperate" or "seperite," but they are certainly better than the word "scramble.")
3. Write a program that asks five questions for which the responses will be words that have more than one acceptable spelling. Give credit for either.
4. Modify the previous program by giving credit for synonyms for the desired responses.

Assignment for More Advanced Students:

1. Write a routine that will compare each character of the INPUT string with characters of a desired response. If 80% of the characters are the same and in the same order, credit a correct response.

Topic 21

CREATING PROGRAMS THAT ALLOW FOR USER ERROR IN ALL INPUTs

Performance Objectives: Students will be able to:

- . write programs that will avoid error messages in user INPUT in the following areas:

arabic numerals and words when only numerals are expected
words when arabic numerals are expected
dollar signs when arabic numerals are expected in money problems

Vocabulary: New: VAL

Review: INPUT, MID\$, LEFT\$, RIGHT\$, INKEY\$ or GET\$

Motivation:

- . What is wrong with these user responses?

HOW MANY CHILDREN DO YOU HAVE? 4 children
HOW MUCH ALLOWANCE DO YOU RECEIVE WEEKLY? \$5
HOW MANY DWARFS DID SNOW WHITE VISIT? SEVEN
HOW MANY BYTES IN 1K MEMORY? 1,024

Aim: How can programs be created that allow for human error in INPUT response statements.

Development:

1. Write a program module that would ask the user how many children he or she has.

Acceptable answers:

10 INPUT "How many children do you have? ";C

or

10 PRINT "How many children do you have?"
11 INPUT C

2. What kind of response would the user have to make in order to avoid a REDO message. (numeric)
3. Introduce VAL as a string manipulation function that will convert the first set of numeric characters in a string into a numeric value and assign the numeric value to a numeric location, ignoring the following alphabetic characters. The value of an alphabetic is 0. The only nondigits permitted in a string are the symbols + (plus), - (minus), and . (decimal point).

C = VAL(C\$)

C is assigned the value 4 if C\$ = "4 CHILDREN"

C is assigned the value 0 if C\$ = "FOUR CHILDREN"

4. Rewrite the program in step 1 with a string variable and VAL to eliminate the possibility of a REDO error message if the response is "4 children."
5. Computer novices often do not understand the REDO error message. How can we use VAL to evaluate a response that we expect to be numerical but instead is alphabetic and print an appropriate message to the user to retype the answer using arabic numerals? For example suppose the user responds to "HOW MANY CHILDREN DO YOU HAVE?" with "FOUR" or "I HAVE 4."

Acceptable answer:

```
15 IF VAL(C$) = 0 THEN PRINT "Please type in numbers only.": GOTO 10
```

6. Rewrite the PRINT "HOW MANY DWARFS DID SNOW WHITE VISIT?" module to accept answers "7", "seven", and "7 dwarfs" as correct, to tell whether incorrect answers are too low or too high, and to avoid a REDO message if the user types in a word answer.

Acceptable answer:

```
10 PRINT "HOW MANY DWARFS DID SNOW WHITE VISIT?"
20 INPUT D$
30 IF D$ = "SEVEN" THEN 100
40 D = VAL(D$)
50 IF D = 7 THEN 100
60 IF D = 0 THEN PRINT "TYPE IN NUMBERS ONLY!":GOTO 10
70 IF D < 7 THEN PRINT "YOUR GUESS IS TOO LOW!":GOTO 10
80 PRINT "YOUR GUESS IS TOO HIGH!":GOTO 10
100 PRINT "YOU'RE ABSOLUTELY RIGHT. THERE WERE ";C$;"."
110 END
```

7. People often respond to questions about money with a dollar sign preceding a number. Using MID\$ or LEFT\$, write a statement that will accept an answer of \$5 and convert it to 5.

Acceptable answer:

```
10 PRINT "How much allowance do you receive?"
20 INPUT D$
30 IF LEFT$(D$,1) = "$" THEN GOSUB 1000
40 D = VAL(D$)
    . . .
50 END
1000 L = LEN(D$)
1010 D$ = RIGHT$(D$,L-1)
1020 RETURN .
```

8. The ways of circumventing an EXTRA IGNORED message when a comma is inserted by the user and the number is greater than 999 can be handled by the ASC function we will study soon. In the meantime, you could include a reminder in your PRINT statement that commas should be omitted when you expect a response large enough for the user to insert one. A comma

can also be handled by building a string one character at a time and checking it for commas.

Applications:

1. Write a program that will ask a user to guess how many states in the United States have names that start with the letter N.
2. VAL can be used to convert INKEY\$ and GET\$ A\$ responses to numeric values to help with multiple choice response evaluation. Write a program that will ask the user to identify the only 20th-century President who did not attend college. (The answer is Harry S Truman.) Include names of three other 20th-century Presidents as wrong answers.

Acceptable answer:

```
100 A$ = INKEY$ or 100 GET$ A$
110 IF A$ = "" THEN 100 ELSE A = VAL(A$)
120 IF A<1 or A>4 THEN PRINT "TYPE IN 1,2,3,OR 4":GOTO 100
```

Summary:

1. What function can determine the arithmetic number in a string that contains a combination of numbers and words?
2. What function will help eliminate a dollar sign from the beginning of a string?
3. How can we avoid a REDO message if a user INPUTs a word instead of a number?

Assignment:

1. Write the following programs. Be sure to write programs that take into account common human response errors.
 - (a) A program that asks the user to guess how many U.S. states have two-word names
 - (b) A program that asks the user to guess how many months there are in a year
 - (c) A program that asks how much money the President of the United States earns
 - (d) A multiple-choice question that asks which planet has the most moons
 - (e) A program that asks the user to guess to the nearest million the current population of the United States.

Assignments for More Advanced Students:

1. Ordinarily the INKEY\$ or GET\$ function can be used only with one-digit numbers or single letters. Write a routine that will use INKEY\$ or

GET\$ and VAL to create strings that have two digits without having to press the <CARRIAGE RETURN> key.

2. Write a subroutine using the INKEY\$ or GET\$ function that will evaluate each entry of a user's response and build a string that contains no commas so that, for example, 1,024 becomes 1024.

Topic 22 ERROR TRAPPING

Performance Objectives: Students will be able to:

- . use the ONERR statement to control errors
- . use error trapping routines to prevent errors
- . learn how to recover from errors and continue program flow

Vocabulary: New: ONERR, error trapping
Review: ASC, GOTO, GOSUB

Motivation:

- . What kinds of actions by the user will cause a computer to generate error messages?

Aim: How can error trapping routines make a program work better?

Development:

1. While detecting and trapping every possible error is a worthy objective, it is rarely achieved. However, we can anticipate many of the frustrating problems that can occur.
2. Some error trapping routines involve programming lines such as those indicated below.

- . How can we prevent an incorrect response to an INPUT?

```
10 INPUT "WHAT IS YOUR NAME"; N$  
20 IF ASC(N$) < 65 THEN 10
```

Explain how this would prevent a number from being entered instead of a name. (The ASC command returns the ASCII value of the first character in a string, and the ASCII value of all numbers is less than 65.)

- . How can we prevent an incorrect number of digits or characters from being entered?

Acceptable answer:

```
10 INPUT "WRITE A FIVE LETTER WORD FOR A PLACE TO LIVE"; AN$  
20 IF LEN(AN$) < > 5 THEN 10
```

3. Virtually all microcomputers have error detection built into their languages. For example, when a command is misspelled, the message "SYNTAX ERROR" appears on the screen of most computers. However, when an error occurs during the execution of a program with no error trapping, the program comes to a halt.
4. In order to control a computer in an error condition, we can begin a program with:

```
ONERR GOTO #
```

When an error condition occurs, the computer will branch to the line number indicated instead of interrupting execution.

5. Study this program:

```
5 REM PROGRAM A
10 ONERR GOTO 50
20 INPUT "PLEASE TYPE YOUR NAME"; N$
30 IF ASC (N$) < 65 THEN 50
30 PRINT "GOOD JOB ";N$
40 END
50 HOME:PRINT "DOES NOT COMPUTE"
60 GOTO 10
```

. If the user types the carriage return without a name many machines will consider this an error and switch control to line 50. The machine will enter an infinite loop and will repeatedly execute lines 5, 10, 50, 60, 5, 10, 50, 60, ...

6. Error trapping procedures must account for the full range of possible responses that one might reasonably expect from a user. The user should not be aware of their existence.
7. A GOSUB statement can be used instead of the GOTO. For example:

```
ONERR GOSUB 50
```

Remember to end the subroutine with a RETURN statement.

Note: To disable the ONERR command in the APPLE computer the following statement is used:

```
POKE 216, 0
```

After this POKE statement is used, the computer will revert to its own error generating routines and halt execution of a program when an error condition exists.

On some other machines the command is ONERR GOTO 0. There are still other techniques for some other computers.

8. Can we distinguish between types of errors when we are in an error trapping routine that was entered via an ONERR command? (Yes. For the APPLE an error code number is generated and kept in memory location 222. For other machines other locations hold the error number code.)

Applications:

1. Look at your reference manual to discuss the different types of error codes and when they are generated.

2. Write a program that will read data statements and continue operating even after attempting to read past the last data statement.

Acceptable answer:

```
5 REM PROGRAM C
10 ONERR GOTO 60
20 FOR I = 1 TO 10
30 READ A$(I)
40 NEXT I
50 END
60 (Continue with program)
```

. . .

```
1000 DATA "JOHN", "MARY", "PETE", "SUE"
```

Summary:

1. What is the purpose of error trapping routines?
2. How do we use the ONERR command to prevent the halting of a program?
3. How we can control answers to an INPUT question?

Assignments:

1. Write a program that will allow a user to INPUT only a three-digit number.
2. Write a program that will only allow a user to INPUT a word longer than seven characters.
3. Write a program that will ask the user to input two numbers, the first of which will be divided by the second. Error trap the program so that if the second number is zero the program will not halt.

Assignments for More Advanced Students:

1. Develop an error trapping routine that will reject a number anywhere in a string.
2. Modify Program A in Development 5 so that the computer will ask for a person's name, age, and weight. Error trap each of these INPUTs so that the program will be "bomb-proof" even to someone who is -13 years old and weighs 0 pounds!

Topic 23

MENUS FOR USER-FRIENDLY CONTROL

Performance Objectives: Students will be able to:

- . explain the advantages of a menu-driven program
- . utilize a menu-driven program for turnkey operation
- . define "user-friendly"
- . implement a simple error trapping routine

Vocabulary: New: user-friendly, menu
Review: INPUT, GET, INKEY\$

Motivation:

- . Restaurant A is in competition with Restaurant B. In order to charge less per meal, Restaurant A decided to spend no money printing menus for customers and to pass the savings along to the customer. Do you think this is a wise decision? Why? (Elicit that a menu is a convenience for the customer in order to assist in making a decision.)
- . If you were judging the quality of educational software, list several criteria that you would look for in order to evaluate a good piece of software. (Answers will vary, but certainly ease of use of a menu-driven program must be included.)

Aim: How can we use a menu routine in a programs?

Development:

1. Explain that "user-friendly" refers to the ease with which a user can take advantage of a program or device.
2. Discuss and develop the program below. What will this program do? (It will list five states and number them 1 to 5. If we type in a number from 1 to 5 the computer will tell us the capital of the state we selected.)

```
10 REM   A MENU DRIVEN PROGRAM FOR CAPITALS OF SOME STATES
20 HOME
30 PRINT "CHOOSE THE NUMBER OF THE STATE AND I WILL TELL YOU ITS
   CAPITAL:"
40 PRINT "1) NEW YORK"
50 PRINT "2) FLORIDA"
60 PRINT "3) CALIFORNIA"
70 PRINT "4) ILLINOIS"
80 PRINT "5) TEXAS"
90 INPUT X
100 IF X = 1 THEN PRINT "THE CAPITAL IS ALBANY"
110 IF X = 2 THEN PRINT "THE CAPITAL IS TALLAHASSEE"
120 IF X = 3 THEN PRINT "THE CAPITAL IS SACRAMENTO"
130 IF X = 4 THEN PRINT "THE CAPITAL IS SPRINGFIELD"
140 IF X = 5 THEN PRINT "THE CAPITAL IS AUSTIN"
150 END
```

3. The menu-driven program will end if a careless end user presses the wrong key. How can we prevent this? (Replace line 150 with: 150 GOTO 90)
This of course creates other problems.

. This is one simple technique for error trapping.
4. How can we modify our menu-driven program so that the moment a number is pressed the computer will branch to the correct response? (Use GET or INKEY\$ statement.)
5. Discuss the term "turnkey programming" and how it applies to computer programming. Define "turnkey programming" as the ability of one program to run other programs.
6. How could we modify our menu-driven program so that, instead of numbers, we can use the letters of the alphabet for user input? How must we change the INPUT statement? (Use a string variable such as X\$ instead of X.)
7. What statements could we add to our menu-driven program to give the user an option to go back to the menu?

Acceptable answer:

```
150 PRINT "WOULD YOU LIKE TO TRY AGAIN (Y/N)?"
160 GET A$
170 IF A$ = "Y" THEN 20
180 IF A$ < > "N" THEN 160
```

- . Why is statement 180 a better programming statement than just END?
(It provides error trapping in case an incorrect key is pressed.)
8. Another use of a menu driver is to allow the user to select from among many programs which are located on the same disk. Essentially one program will simply run another on the same disk.

Summary:

1. What is menu-driven programming?
2. Why do we use menus in some programs?
3. How may we define the terms "user-friendly" and "turnkey programming"?
4. What are the advantages of error trapping?

Assignments:

1. Write a menu-driven program that will list five countries numbered 1 to 5. If the user types a number from 1 to 5, the computer should tell the capital of that country. Be sure to error trap your program.

2. Write two small programs, each able to run the other. Name each program in a REM statement as it would appear on disk, and use the proper disk operating system (DOS) control for your machine.
3. Write the programming statements that could be added to the menu-driven program discussed in this lesson so that the computer would ask, "Are you sure you want selection 1?" Thus, whatever menu number you selected, the computer would supply a user option allowing a change of mind.
4. Discuss what programming techniques you would use to make a menu more attractive.

Assignments for More Advanced Students:

1. Write a menu-driven program that causes all other selections to disappear from the screen once a choice is made.
2. Write a menu driven program that would quiz a user on the capitals of the 50 states.

Topic 24
**INTERNAL DOCUMENTATION FOR USER (PRINTs)
STRESSING UNAMBIGUOUS LANGUAGE**

Performance Objectives: Students will be able to:

- . write PRINT statements requiring string responses
- . write programs using INKEY\$ and GET\$
- . write programs allowing for multiple responses

Vocabulary: New: user-friendly, INKEY\$, GET

Review: INPUT, PRINT, ON ___ GOTO ___, ON ERROR GOTO ___,
documentation, CAI

Motivation:

- . What statements could we write in BASIC to tell people to type their names into a computer?

Acceptable answer:

```
10 PRINT "TYPE IN YOUR NAME."  
10 PRINT "TYPE IN YOUR NAME AND PRESS THE <CARRIAGE RETURN> key."
```

- . Stress the appropriate situation for using each response. (More explicit instructions have to be written for less experienced computer users.)

Aim: How can we write programs that are simple for a user to respond to or interact with?

Development:

1. Have the students write statements requesting the user to tell whether he or she wants to continue with the program.

Acceptable answer:

```
100 PRINT "TYPE IN 'YES' IF YOU WANT TO CONTINUE."  
110 PRINT "TYPE IN 'NO' IF YOU WANT TO STOP."  
120 PRINT "PRESS <CARRIAGE RETURN> WHEN FINISHED."  
130 INPUT YN$  
140 IF YN$ = "YES" THEN 20  
150 IF YN$ = "NO" THEN 999  
160 PRINT "PLEASE FOLLOW THE INSTRUCTIONS!"  
170 GOTO 100
```

- . Reinforce the fact that lines 140 to 170 could be replaced by the following statement if people didn't make errors and if the computer allowed the ELSE statement.

```
140 IF YN$ = "YES" then 20 ELSE 999
```

- . Rewrite the request to continue or not using the INKEY\$ or GET\$ functions.

- . Use the INKEY\$ or GET\$ function to gather the response to the message asking whether the user wants to continue.

Acceptable answer:

```
100 PRINT "Type in 'Y' if you want to continue."
110 PRINT "Type in 'N' if you want to stop."
120 LET YN$ = INKEY$ (or on some machines 120 GET YN$ )
130 IF YN$ = "" THEN 120
140 IF YN$ = "Y" THEN 20
150 IF YN$ = "N" THEN 999
160 GOTO 100
```

- . What other line could the program branch to in line 160? (It could branch to 120, and then the instructions would not be reprinted.)
 - . How could lines 140 to 160 be rewritten in IF ____ THEN ____ ELSE ____ form?
(140 IF YN\$ = "Y" THEN 20 ELSE 999)
2. Have the students write a CAI multiple-choice program module that asks, "Who was the sixteenth President of the United States?"

Acceptable answer:

```
100 PRINT"WHO WAS THE SIXTEENTH PRESIDENT"
110 PRINT"OF THE UNITED STATES?"
120 PRINT
130 PRINT " (1). GEORGE WASHINGTON"
140 PRINT " (2). ABRAHAM LINCOLN "
150 PRINT " (3). HARRY S TRUMAN"
160 PRINT " (4). DWIGHT D. EISENHOWER"
170 ON ERROR GOTO 500
180 INPUT X
190 ON X GOTO 200,250,300,350
200 PRINT "SORRY, GEORGE WASHINGTON WAS #1."
210 PRINT "      LINCOLN WAS #16."
220 GOTO 999
```

* * *

- . What is the purpose of line 170? (prevents the program from being stopped on an error and returns control to line 500)
- . What is the purpose of line 190? (branches to line 200 if X = 1, to 250 if X = 2, 300 if X = 3, 350 if X = 4)
- . What routine might be printed starting at line 250?

Acceptable answer:

```
250 PRINT " Right, Lincoln was #16."
260 GOTO 999
```

- . Have the student print appropriate routines for the other two possible choices.
- . What routine might be found at line 500?

Acceptable answer:

```
500 PRINT " Type in 1 or 2 or 3 or 4"
510 PRINT " and press enter."
520 RESUME 180
```

Applications:

1. Write a program that will allow for the responses "Always", "Sometimes", and "Never".
2. Rewrite the program above with INKEY\$ or GET\$ and "A", "S", and "N".
3. Write a program for a multiple choice question on "Who is the president of our student organization?"
4. Rewrite the above program with an INKEY\$ or GET\$.

Summary:

1. What factor determines how explicit you have to be in a print statement preceding an INPUT?
2. What is the advantage of INKEY\$ or GET\$?

Assignments:

1. Write a program subroutine that will branch to line 100 if the INPUT is true or line 20 if the INPUT is false.
2. Rewrite the above program with INKEY\$ or GET\$.
3. Write a program subroutine that will allow the user to go back to the very beginning of a program (Instructions), to go to the middle (Restart game), or to exit a program (END).
4. Write a program that will ask a user to guess how many months in the year have the letter r in them. Allow for four incorrect guesses before the machine prints out the months that have an r in them.
5. Write a program that will ask a user to type in the number of siblings he or she has. Include appropriate comments.

Topic 25
EXTERNAL DOCUMENTATION IN THE FORM OF
PRINTED DESCRIPTIVE MATERIAL

Performance Objectives: Students will be able to:

- . list three reasons why external documentation might be used instead of internal documentation
- . name five components of external documentation
- . write external documentation for a program

Vocabulary: New: external documentation

Review: internal documentation, PRINT, REM

Motivation:

- . Why are you given a manual or booklet when you buy a DOS disk or a complicated game? (Complicated instructions and the procedures, nature, and purposes of special features would take up too much space on the disk.)

Aim: What is external documentation for a computer program?

Development:

1. If possible, load a COBOL, FORTRAN, or Pascal disk or a complicated game disk new enough so that the class members are not already good at playing it. You need something to recapture the frustration of having a program but not being able to use it because you do not have the instructions.
2. After it becomes obvious that you need instructions on how to use the software, have students look through a pile of manuals (if possible) for the right brochure. It would be ideal if you had manuals for several makes of machine for the same program, for example Pascal for the Apple and Pascal for TRS-80.
3. Elicit from pupils the essential components of external documentation:
 - . title of software it accompanies
 - . make and model of machine for which the accompanying disk is designed
 - . introduction or overview
 - . getting started
 - . running the program
 - . technical information (if needed)
 - . appendices or glossary for special information (if needed)
 - . index (if needed)
4. Have pupils separate into groups and write the external documentation for a program that will count the number of one-letter words, number of two-letter words, number of three-letter words, etc., in a given INPUT string. Have one group write for a TRS-80, one for a PET, one for an APPLE, one for an ATARI, and so on. One group might write the program for a 16-bit-per-byte machine so you can discuss changes necessary for that machine.

- . Have students create a title and decide the make and model of machine the program is intended for.
- . Problem #1: Our 8-bit-per-byte machines have a maximum string length of 255. The standard paragraph is larger than that. How do we overcome that problem? (Use INKEY\$ or GET\$ to input the letters one character at a time. Create a string. Employ a counter to keep track of the number of characters being INPUT. When the string becomes large, students will put subsequent characters in additional strings. Students will have options to decide what "large" is. Do they want to stop at a space after, possibly, 240 characters, or will they continue until 255 are contained in the string?)
- . Problem #2: How are we going to deal with hyphenated words, words like A#1 that are alphanumeric mixtures, or the #2 that we see above in "Problem #2"? The class should make a decision and then make a note to include that information in the external documentation.
- . The program could do many other things, such as count the number of occurrences of a specific vowel or character. Each group should use knowledge of the program's capabilities to create options to be included in the documentation.
- . Save at least the last 10 minutes of the period so each group can send a representative to the board to write a panel of what would be in that group's brochure.

Summary:

1. Why is external documentation sometimes necessary?
2. What should be found in the printed descriptive material that accompanies a program?

Assignments:

1. Write external documentation for a moving graphics program that would enable the user to create different graphics while using your program. Modify a previously developed program.
2. Develop a game program using graphics and, if possible, sound. Write external documentation for it.
3. Write a CAI program for schoolmates with remedial needs. Develop external documentation to accompany this CAI project.

Assignment for More Advanced Students:

1. Code and write external documentation for an antonym program that allows the user to use a given set of antonyms or to add new sets of antonyms for another user.

Topic 26

IMPROVING USER-FRIENDLINESS BY EXTENDING INPUT SCOPE TO ALLOW TOLERANCE FOR USER RESPONSES THAT ARE CLOSE BUT NOT EXACT

Performance Objectives: Students will be able to:

- . write any equality as a BASIC statement allowing for tolerance
- . write any inequality as a BASIC statement allowing for tolerance
- . list which arithmetic operations are more likely to lead to a calculation discrepancy

Vocabulary: New: tolerance, test data

Review: ABSolute value, <, >, single precision, double precision

Motivation:

- . You have written a program using division and square root functions and the answer is printed as 1.999999. You expected the answer to be 2. What might account for the error?
- . Elicit from the class the reasons why there might be a discrepancy between a computer result and a pencil and paper result. (for example, change to and from base 2, rounding off after a certain number of decimal places)

Aim: How do we correct programs to allow for computer calculations that yield answers that disagree with correct answers because of roundoff error or conversion from base 10 to base 2 errors?

Development:

1. Review the concept that computers do not work in the decimal (base 10) system, but rather in the binary (base 2) or hexadecimal (base 16) system.
2. Review the concept of truncation, or "chopping off," of places after a certain number of decimal places is reached. That is, $2/3 = .6666667$ or whatever degree of precision is appropriate to the machine you use.
3. List instances when high accuracy is not needed. (such as when all the values are integers and the four basic operations of addition, subtraction, multiplication, and division are used) Most of our work is with these operations. We must compensate for tolerance, or margin of error, only when our programs become more complex.
4. Degree of tolerance depends on the method of computation and whether we are working with single or double precision numbers.
5. 100 IF A = B THEN 150

becomes

100 IF ABS(A-B) < .00001 THEN 150

Subtracting one side of an equality from the other side should give you a tiny tolerance. The ABSolute value is needed because the difference might be positive or negative. A tolerance of .00001 works well with single precision numbers.

6. Formula for inequalities and tolerance:

```
100 IF A<>B THEN 150
```

becomes

```
100 IF ABS(A-B) > .00001 THEN 150
```

Applications:

1. Rewrite each statement to allow a tolerance of .00001

(a) 100 IF A/B = C/D THEN 200

(b) 50 IF (3*R)/((1/3)) = .4 THEN 75

(c) 30 IF C = SQR(A[2+B[2]) THEN GOSUB 500

(d) Rewrite (a) through (c) above with inequalities rather than equalities

Summary:

1. Why is there a difference between some computer calculations and paper and pencil calculations?
2. What circumstances are likely to generate these differences?
3. What substitution might we make for a statement like:

```
10 IF A = B then 100
```

4. What formula will take into account tolerance in a statement of the form:

```
10 IF A<>B THEN 100
```

Assignments:

1. Change the following statements to allow for tolerance.

(a) 40 IF (B[2 - 4*A*C])/((1/2)) = 0 THEN PRINT "Roots ="

(b) 40 IF (B[2-4*A*C])/(.5)<>0 THEN PRINT "Roots not ="

(c) 40 IF (C[2 = A[2 + B[2] THEN 300

2. Write BASIC statements for the formulas at the top of the next page, using the number in parentheses as the tolerance.

$$1) \text{ Potential Energy} = \text{mass} \times \text{acceleration due to gravity} \times \text{height} : PE = mgh \quad (.0001)$$

$$2) \text{ Density of a substance} = \text{mass} / \text{volume} : D = m/v \quad (.00001)$$

$$3) \text{ Relative humidity (\%)} = \frac{\text{vapor pressure of dewpoint temp}}{\text{vapor pressure of dry-bulb temp}} \times 100 \quad (.0001)$$

$$4) \text{ Eccentricity of an ellipse} = \frac{\text{distance between foci}}{\text{length of major axis}}$$

$$5) \text{ Profit} = I - 8T + 12T - C \text{ where } T = \text{time in years, } C = \text{cost in thousands} \quad (.001)$$

$$6) \text{ Effective annual interest rate} = \left(\frac{\text{future value}}{\text{initial investment}} \right)^{\frac{1}{\text{years}}} - 1 \quad (.00001)$$

3. Write a program that will print the numbers from 1 to 20 along with each number's square root using the fraction form (1/2) and decimal form (.5), fourth root, fifth root, and eighth root, all in both forms.

Topic 27

EASILY UNDERSTOOD DESCRIPTIONS (PART I)

Performance Objectives: Students will be able to:

- . provide PRINTed statements that are both pleasing to look at and user friendly
- . format output so that results can be easily read, and insure that no pertinent information will be overlooked
- . use the comma in conjunction with the PRINT statement to achieve neat formats
- . use the semicolon in conjunction with the PRINT statement for a readable horizontal output
- . develop skill in using the PRINT TAB() statement
- . understand the use of the PRINT USING statement
- . write BASIC programs to simulate PRINT USING for computers that do not support this command

Vocabulary: New: PRINT USING, PRINT TAB()

Review: LEN(), MID\$, STR\$, VAL(), user-friendly programs, screen formatting

Motivation:

- . When writing a BASIC program we develop a plan, formulate a flowchart and collect our algorithms (solution patterns). Are there any other considerations before we begin our coding? (Consider the user who is, in most cases, unable to write programs.)
- . What are some ways to design programs so as to assist users in understanding the input and output of a program?
- . What idiosyncrasies of computer output might be confusing to an inexperienced user?

Acceptable answers:

- Output scrolls off screen.
- There is no automatic dollar and cents format.
- Numbers are left-justified and decimal points are not aligned.
- Something as simple as "press return to continue" could be confusing to a user that presses the wrong key.

Aim: What are some of the ways we can PRINT our output so that it is more easily understood and user friendly?

Development:

1. Describe the output of the following program:

```
10 FOR X = 1 TO 10
20 PRINT X,2X,X + 5
30 NEXT X
40 END
```

101

The use of the comma with the PRINT statement is a valuable formatting tool. It enables the results of a program to be printed in neatly spaced columns. The number of columns allotted depends on the computer you are using. What are some of the things we must take into consideration when using this format?

- . The width of the column. A 40-column screen divided evenly into three columns provides for a maximum column width of 13 characters. On an 80-column screen, five columns would allow a maximum of 16 characters in each column.
 - . The type of data that is being PRINTed in the columns. We have two choices: numeric and alphanumeric.
 - . The length of each column. Some screens provide 24 lines of PRINTed data. The cursor takes one position, column headings take at least one line -- two lines if we skip a line or underline for clarity.
2. The teacher should point out that although the PRINT "," is a very useful tool it has limitations. Student discovery by experimentation will be useful in impressing on students the limitations of this command. First, establish the number of zones the computer supports. Tell students the number depends not only on the computer but on the type of Basic being used.

For example: The Apple computer will provide five zones if Integer Basic is used and only three zones with Applesoft. Most CBM computers provide three zones of 10 characters each. The number of zones and characters per zone should be determined by experimentation. Use immediate mode as well as program mode.

A simple program like the one that follows will show students the way numbers are converted to exponential form when they exceed the required number of characters per zone and how zones are eliminated when the number of characters overwrite the next zone.

```
10 FOR X = 100 TO 300 STEP 100
20 PRINT X*X,X*X*X,X*X*X*X
30 NEXT X
40 END
```

Experiment with strings. The Apple Computer will accept an eight-character number but only a seven-character string.

3. Next, draw student's attention to the use of the semicolon with the PRINT statement. Establish the horizontal nature of the PRINT when a semicolon is used. Demonstrate a quick way of drawing a horizontal line for the purpose of underlining titles:

```
10 FOR T = 1 TO 40
20 PRINT "-";
30 NEXT T
```

- Change the "-" to "=" or "*" for emphasis. Cover the entire screen with stars by changing line 10 to:

```
10 FOR X = 1 TO 840 (or a different number for another computer)
```

- Discuss with students how the semicolon suppresses the carriage return. If your system has cursor movement keys, write a program that will PRINT HELLO in the tenth row and fifteenth column.

```
10 PRINT "<CLS>"
20 FOR H = 1 TO 10
30 PRINT "<DOWN CURSOR>";
40 NEXT H
50 FOR H = 1 TO 14
60 PRINT "<RIGHT CURSOR>";
70 NEXT H
80 PRINT "HELLO"
```

Otherwise use PRINT CHR\$(10) for a down cursor and PRINT " " for a right cursor. Tell students that the semicolon is sometimes used to eliminate spaces between numbers and strings. For example:

```
10 B$ ="DOLLAR TICKETS"
20 FOR A = 10 TO 50 STEP 10
30 PRINT A;B$
40 NEXT A
50 END
```

```
10DOLLAR TICKETS
20DOLLAR TICKETS
30DOLLAR TICKETS
40DOLLAR TICKETS
50DOLLAR TICKETS
```

- Ask students to predict the results of placing a comma after the "\$" in line 30. Let them experiment to see if they predicted correctly.

Note: Since this is a 16-character string, chances are that two nonaligned strings will be PRINTed per line unless your machine allows 16-character columns.

4. Introduce the PRINT TAB() statement. This statement works in a similar fashion to the tabulator on a typewriter. The expression inside the parentheses is either a number or a variable that represents a number. PRINT TAB(X) cannot be used independently but must be included in a PRINT statement. There can be several TABS with one PRINT, and most computers require no punctuation other than the "(" to enclose the number. For the PRINT TAB() the screen lines are numbered 1 to 80 (or less than 80, depending on your machine's configuration) and the result of PRINT TAB(8) "*" is to PRINT a "*" in the eighth screen column.

Have students use the PRINT TAB function to format the output of the following program.

- . Find the perimeter of regular polygons having sides of 10 to 25 units.
- Page title should be POLYGON PERIMETERS. Column titles are SIDE, TRIANGLE, SQUARE, PENTAGON, HEXAGON.
- Page title contains 18 characters. To space evenly, tab to leave 11 spaces on each side.
- Underline page title for emphasis.
- Column titles contain 33 characters. We cannot leave two spaces after each title if our machine displays only 40 columns, as that would require 41 columns. If your machine allows more than 40 columns, adjust accordingly. It is best to skip only one space after SIDE and then skip two spaces after each of the other column titles.
- Numbers will have to be right-justified. Review changing numeric variables to strings and concatenating with " ".

Suggested program:

```

10 REM POLYGON PERIMETERS
20 HOME (or CLS)
30 PRINT TAB(11)"POLYGON PERIMETERS"
40 FOR X = 1 TO 40
50 PRINT "-";
60 NEXT X
70 PRINT
80 PRINT "SIDE";TAB(6);"TRIANGLE";TAB(16);"SQUARE";TAB(24);
  "PENTAGON"; TAB(34)"HEXAGON"
90 FOR P = 10 TO 25
100 A = 3 * P:B = 4 * P:C = 5 * P:D = 6 * P
110 A$ = STR$ (A):B$ = STR$ (B):C$ = STR$ (C):D$ = STR$ (D)
120 IF LEN (B$) < 3 THEN B$ = " " + B$
130 IF LEN (C$) < 3 THEN C$ = " " + C$
140 IF LEN (D$) < 3 THEN D$ = " " + D$
150 PRINT P TAB(8)A$ TAB(16)B$ TAB(25)C$ TAB(35)D$
160 NEXT P
170 END

```

5. Teach the PRINT USING statement. Tell students that even though all forms of Basic do not support this statement, programs that simulate this command are easily written. The PRINT USING statement does the following:

- . Causes a table of numbers to be written so that the decimal points are aligned
- . Permits the programmer to determine the number of decimal places, and then all numbers will be rounded off to that place
- . Permits punctuation to be included with the number -- for example, the dollar sign, the comma, the colon
- . Ensures that all numbers are written with a fixed decimal point format -- for example, 14.00, not 14

Note: Check your machine's PRINT USING format. All machines differ slightly in their syntax.

6. Compare the following programs and program output.

Program:

10 DATA 2,2.5,2.555	10 DATA 2,2.5,2.555
20 FOR N = 1 TO 3	20 N\$ = "###.##"
30 READ N	30 FOR P = 1 TO 3
40 PRINT N	40 READ P
50 NEXT N	50 PRINT USING N\$;P
60 END	60 NEXT P
	70 END

Output:

2	2.00
2.5	2.50
2.555	2.56

- . Notice the change in alignment. Notice the way the 2.555 is rounded. The format of the PRINT USING statement is set in step 20. Here we have specified that numbers are to be rounded to two decimal places. How would we specify four places? Line 50 is used to call the statement in line 20. Strings can be written into the PRINT USING statement. The following program is an example of this:

```
10 N$ = " THE SQUARE ROOT OF ### IS ###.##"
20 FOR S = 100 TO 999
30 LET A = SQR(B)
40 PRINT USING N$;B;A
50 NEXT S
60 END
```

```
( THE SQUARE ROOT OF 100 IS 10.00
  THE SQUARE ROOT OF 101 IS 10.05
  etc. )
```

7. Now let us write a Basic program to simulate this very useful function. Review the following commands: LEN(), MID\$(), STR\$(), VAL(). Write a program to simulate dollar-and-cents format. Round all decimals to two places.

Suggested program:

```
10 REM PRINT USING SIMULATION
20 INPUT R$
30 R = LEN(R$)
40 FOR T = 1 TO LEN(R$)
50 IF MID$(R$,T,1) = "." THEN Q = T: GOTO 80
60 NEXT T
70 R$ = R$ + ".00": GOTO 120
80 IF R - Q = 1 THEN R$ = R$ + "0": GOTO 120
```

105

```

90 IF R - Q > 2 THEN N = VAL (R$)
100 N = INT ((N + .005) * 100) / 100
110 R$ = STR$ (N)
120 PRINT R$
130 END

```

Input: 2 2.2 2.22 2.222 2.225

 (2.00 2.20 2.22 2.22 2.23)

Applications:

1. The formula for converting Fahrenheit temperatures to Celsius is $C = (5/9) * (F - 32)$. Write a program to produce the following table in increments of 10 degrees Celsius.

CELSIUS	FAHRENHEIT
0	32
10	50
20	68
.	.
.	.
.	.
100	212

2. Write a program that will use the "," in conjunction with PRINT to produce the following output.

RECTANGLE AREAS =====

LENGTH	WIDTH	AREA
-----	-----	-----
40	11	440
40	12	480
40	13	520
50	11	550
50	12	600
50	13	650
60	11	660
60	12	720
60	13	780

3. Use immediate mode to determine the way the computer treats a PRINT TAB(0). Replace the 0 with a number greater than 40. Predict the results of using a number greater than 255.
4. Determine the value of X in PRINT TAB(X), that will center the following strings on the video screen:
 - (a) PROGRAM LOADING
 - (b) CALCULATING ELAPSED TIME
 - (c) THINK!
 - (d) LET'S GO
 - (e) IMPORTANT DATES TO REMEMBER

5. Determine the output for the following program:

```
10 DATA 6.87,9.353,2
20 P$ = "#.##"
30 FOR K = 1 TO 3
40 READ X
50 PRINT USING P$;X
60 NEXT K
70 END
```

6. Write a program that will produce the following output. Use the PRINT USING statement.

```
THE SQUARE OF 10 IS 100
THE SQUARE OF 15 IS 225
THE SQUARE OF 20 IS 400
THE SQUARE OF 25 IS 625
THE SQUARE OF 30 IS 900
```

Summary:

1. What are some benefits of displaying outputs in orderly tabular arrays?
2. Why are titles a necessary part of user-friendly output?
3. Can a 10-character number be displayed in the fourth zone of a computer that provides four zones when the PRINT "," instruction is used?
4. What would be an efficient way of covering the entire screen with the character "X"?
5. Describe the difference between the formatting of the output when PRINT is used alone, with a comma, with a semicolon, and with a TAB(#).
6. Why is it important to know the size of the screen when writing a user-friendly program?
7. How does the computer justify numbers?
8. What are the problems with this type of justification?
9. Can programs be written that change this type of justification?
10. When is it expedient to write 5.00 instead of 5?
11. Are there times when it is useful to use the cursor control keys in a Basic program?

Assignments:

1. Write a program that will PRINT GOOD-BYE in the middle of the screen.

Make use of the following keys in your program. "CLS" (or similar statement), ";", "<down cursor>", "<right cursor>"

2. In a dance contest there are eight couples and four judges. Each judge rates the contestants on a scale of 1 to 10. Write a program that will determine the winning couple. Format the results neatly on the screen. (Suggestion: use five vertical columns, four for each judge and one for the totals. Use eight horizontal columns, one for each couple, and PRINT the names of the winning couple on the bottom of the screen.)
3. The two keys "<" and ">" make a nice design when PRINTed together. Write a program that will cover the screen with these two symbols PRINTed consecutively. Make use of the semicolon.

Topic 28

EASILY UNDERSTOOD DESCRIPTIONS (PART II)

Note: For teachers using machines which do not allow VTAB or HTAB instructions, cursor control character printing accomplishes the same goal. Check the operator's manual for your machine's cursor control potential.

Performance Objectives: Students will be able to:

- . use the HTAB and VTAB statements to improve the output of a program
- . write programs to simulate HTAB and VTAB for computers that do not support these instructions
- . perform simple operations on matrices and to print matrices, properly formatted, on the video screen
- . neatly format the screen when a program has a jagged output
- . learn techniques for printing large outputs that cannot fit on one screen
- . make proper use of the pause, the GET statement, and keyboard reading techniques when formatting their outputs
- . use the PRINT SPC() and the POS() commands

Vocabulary: New: HTAB, VTAB, MATRIX, MATRICES, PRINT SPC(), POS()
Review: GET, POKE, PEEK, GOSUB, DIM(), screen formatting, user friendly

Motivation:

- . Once we have written data on a screen line is it possible to return to that line for the purpose of adding additional data?
- . Is it possible to use PRINT TAB() to place additional data in the line?
- . How can we save data in a program for printing at a later time?
- . Assume a program saves five arrays. Each of these arrays contains a different number of elements. Is it still possible to print our output in five neat columns?
- . If our output takes more than 40 screen lines can we provide for our user to see everything before it scrolls off the screen?

Aim: How can we print large, uneven outputs and still keep our programs user friendly?

Development:

1. A multipurpose program is divided into several sections. In an attempt to make the program user friendly, each section is introduced by clearing the screen, framing it with stars, and printing an introduction in the center of the screen. Have students write a program that will introduce the section that tests "FUNDAMENTAL SKILLS".
2. Review the use of the semicolon with the PRINT statement. Teach students

the way that cursor control is accomplished when using the HTAB and VTAB commands. Introduce VTAB and discuss its similarities to "cursor up." Frame the screen with stars using just VTAB and PRINT TAB().

3. Stress the purpose of returning the cursor to the bottom of the screen with VTAB 23 so that it will not overwrite the display. Have students attempt to write in the middle of the screen by returning to a line and TABbing to the center of the screen. Have them see that the PRINT TAB() will erase the star from that screen line.
4. Establish the need for another command and introduce the HTAB. Stress the differences between the TAB and the HTAB and VTAB.

TAB is part of a PRINT STATEMENT	VTAB and HTAB stand alone
TAB requires parentheses	VTAB and HTAB do not
TABS are repeated in one step	V and HTABS are separate statements
TABS require no punctuation	V and HTABS must end with a colon when used in multiple commands

5. Complete the program.

Suggested program:

```

5 HOME
10 FOR X = 0 TO 39
20 PRINT "*";
30 NEXT X
40 C = C + 1
45 IF C = 2 THEN 60
50 VTAB 22: GOTO 10
60 FOR A = 2 TO 21
70 VTAB A: PRINT "*" TAB( 40) "*"
80 NEXT A
90 VTAB 5: HTAB 14: PRINT "THIS PROGRAM"
100 VTAB 7: HTAB 15: PRINT "TESTS YOUR"
120 VTAB 9: HTAB 17: PRINT "SKILLS"
130 VTAB 11: HTAB 19: PRINT "IN"
140 VTAB 13: HTAB 14: PRINT "FUNDAMENTALS"
150 VTAB 23

```

6. Ask students if they think these commands can be simulated using the cursor control keys. Remind them of the way they positioned a word in the center of the screen in the preceding lesson. Write a program that will divide the first 20 integers into lists of even and odd integers using these keys to format the output. Review the DIM() statement and the one-dimensional array. Do not eliminate the zeros at this point. A display containing all the zeros is useful in reminding students the way computers store numbers in arrays. Review the GOSUB statement. This will be useful in shortening the program.

Suggested program:

```

10 PRINT "<CLS>:REM (or your equivalent)"
15 DIM A(20),B(20)

```



```

20 PRINT TAB(10);"EVEN";TAB(20);"ODD"
25 FOR X = 1 TO 20
30 IF X/2 = INT(X/2) THEN A(X) = X
40 IF X/2<> INT(X/2) THEN B(X) = X
45 NEXT X
50 FOR I = 1 TO 20
55 GOSUB 250
60 PRINT A(I)
70 NEXT I
80 FOR I = 1 TO 20
90 PRINT "<CURSOR UP>";
100 NEXT I
110 FOR I = 1 TO 20
120 GOSUB 200
130 PRINT B(I)
140 NEXT I
150 END
200 FOR T = 1 TO 20
210 PRINT "<CURSOR RIGHT>";
220 NEXT T
230 RETURN
250 FOR T = 1 TO 10
260 PRINT"<CURSOR RIGHT>";
270 NEXT T
280 RETURN

```

7. Ask students what would have happened if we attempted to print lists of 100 odd and even numbers on the video screen. Elicit that it is sometimes necessary to divide the output among two or more displays. Develop techniques for this procedure.
 - . The pause, useful only when short periods of time are required
 - . GET G\$, permits the user to read the output at his own pace
8. Have the class determine that the GET or INKEY\$ statements are best for their purposes and review their use. Write a program that divides the first 100 numbers into abundant, perfect, and deficient numbers, depending on the sum of their proper divisors. Since we are using 100 numbers and our screen can display only 24 lines, it is wise to divide our output among four screen displays. Whenever we have multiple screen displays it is always important to repeat the titles. It is also necessary to instruct the user to press any key when he or she wishes to continue.
9. Another user-friendly technique is to inform the user that the program has finished. We can't expect the users to know that no more information is forthcoming unless we tell them that the program is over. Return to the previous program on odd and even numbers. Have students explain how to eliminate the zeros from the printout. Discuss the use of multiple counters. Decide to use an IF = 0 statement.

Suggested program:

```
10 DIM A(100),B(100),C(100)
20 I = 1
30 REM ABUNDANT,DEFICIENT AND PERFECT NUMBERS
40 FOR N = 3 TO 100
50 S = 1
60 FOR D = 2 TO SQR (N)
70 IF N / D = INT (N / D) THEN 90
80 GOTO 110
90 IF N / D = D THEN S = S + D: GOTO 110
100 S = S + D + N / D
110 NEXT D
120 IF S > N THEN A(I) = N
130 IF S = N THEN B(I) = N
140 IF S < N THEN C(I) = N
150 I = I + 1
160 NEXT N
170 A = 1:B = 25
180 HOME (CLS or your equivalent)
190 PRINT TAB(3)"ABUNDANT" TAB(15)"DEFICIENT" TAB(28)"PERFECT"
200 VTAB 23 (or your equivalent): PRINT "PRESS ANY KEY FOR ADDITIONAL
    NUMBERS"
210 VTAB 3 (or your equivalent)
220 FOR T = A TO B
230 IF A(T) = 0 THEN 250
240 HTAB 5: PRINT A(T) (or your equivalent)
250 NEXT T
260 VTAB 3 (or your equivalent)
270 FOR T = A TO B
280 IF B(T) = 0 THEN 300
290 HTAB 30: PRINT B(T) (or your equivalent for HTAB)
300 NEXT T
310 VTAB 3 (or your equivalent)
320 FOR T = A TO B
330 IF C(T) = 0 THEN 350
340 HTAB 17: PRINT C(T) (or your equivalent)
350 NEXT T
360 GET G$:A = A + 25:B = B + 25
370 IF A > 100 THEN HOME : VTAB 10: HTAB 10: PRINT "THAT'S ALL FOLKS!":
    END
380 HOME (CLS or your equivalent)
390 GOTO 190
```

10. Matrices are used often in mathematics and it is good to know how to print them on the screen. A short discussion of the matrix would be useful here, but make no attempt to teach any theory at this time. Once all students know what a matrix is, they should attempt to print in columns and rows with all numbers right-justified. The proper placement of VTAB and HTAB is crucial to this display. Also stress the use of PRINT to begin a new line. Take care that students realize that negative signs are counted as one character. It is also important to leave spaces after each of the elements. Before students attempt to write the program, a brief review of the two-dimensional array would be useful.

Note: Use your machine's equivalents for HOME, HTAB, and VTAB.

Suggested program:

```
1130 REM SAMPLE MATRICES
1140 HOME
1150 PRINT TAB( 5)"THIS IS A SAMPLE OF A 5 X 7 MATRIX"
1160 DIM A(5,7)
1170 FOR R = 1 TO 5
1180 FOR C = 1 TO 7
1190 IF R = 1 THEN A(R,C) = C
1200 IF R = 2 THEN A(R,C) = 2 * C
1210 IF R = 3 THEN A(R,C) = 3 * C
1220 IF R = 4 THEN A(R,C) = 4 * C
1230 IF R = 5 THEN A(R,C) = 5 * C
1240 NEXT C
1250 NEXT R
1260 VTAB 8: HTAB 8
1270 FOR R = 1 TO 5
1280 FOR C = 1 TO 7
1290 IF A(R,C) < 10 THEN PRINT " "A(R,C)" ";: GOTO 1310
1300 PRINT A(R,C)" ";
1310 NEXT C
1320 PRINT : HTAB 8
1330 NEXT R
```

11. This program has its limitations; it works only for two character numbers. Change step 1190 to read -C instead of C. Execute the program and have students see the need to change step 1290 to

```
1290 IF A(R,C) < 10 AND A(R,C) > 0 THEN PRINT " "A(R,C)" ";:GOTO 1310
```

Formatting matrices for more than two characters can be left for an assignment.

12. Another instruction that helps us format our output is SPC(). This instruction permits us to space over a given number of character positions. Like TAB, SPC must be included in a PRINT statement. Unlike TAB, the columns are numbered 0 to 39 for PRINT SPC.

The way this command works depends on the computer you are using. For CBM computers, text that the cursor passes over is not deleted. When this command is used on the Apple, all text written over is erased. Conduct experiments to impress on students the differences between the SPC and the TAB command. A short program that will demonstrate the difference in screen position is:

```
10 PRINT TAB(5)"HELLO"
20 PRINT SPC(5)"HELLO"
```

Note: For those machines without SPC simply PRINT " " instead of PRINT SPC(5).

13. Next, experiments should be conducted to see when text is preserved and when it is deleted. A useful command that enables us to determine the

current column position of the cursor is the POS() statement. When thinking through the formatting of output it is often useful to know where the cursor will be at the end of an instruction.

We get this information by asking for the column position of the cursor with this command. Like SPC(), the screen columns are numbered 0 to 39. The number inside the parentheses is a dummy variable; thus we will use 0. To ask for the current column position of the cursor use the following instruction.

```
10 PRINT "THE CURSOR IS NOW AT COLUMN ";POS(0)
```

```
( THE CURSOR IS NOW AT COLUMN 28. )
```

- . Remind students that the cursor must advance one space after each output.
- . Elicit situations when this technique could save valuable time. By now students must realize that designing the output of a program is both time consuming and tedious. Any techniques that will lighten the burden are truly welcomed.

Applications:

1. Write a program that will list the multiples of 11, 12, 13, 14, and 15 that can be found in the first 150 positive whole numbers. Save the numbers in an array and make the output pleasing to look at.
2. Write a program that frames the screen with % signs and prints CALCULATING in the center of the screen. Remember to position the cursor so it does not interfere with the display.
3. Give the output of the following programs.

```
10 PRINT TAB(10)"CURSOR AT COLUMN ";POS(0)
10 PRINT SPC(10)"CURSOR AT COLUMN ";POS(0)
```

4. The transpose of a matrix is formed by interchanging the columns and the rows. Return to the program that we wrote earlier and explain what change you would make to print the transpose of our 5-by-7 matrix.
5. Generate Pythagorean triples with the Pythagorean formula. Choose values of A and B from 1 to 25. Have computer list A, B, and C for each possible choice and indicate if the result is or is not a triple. Determine how many screen displays are needed. Use the GET statement to access the successive screens. Inform the user when the program is completed.

Summary:

1. What advantage is there to using an HTAB rather than a PRINT TAB?
2. If the cursor is in row 10 to begin with, how can we move it to the top middle of the screen?

3. Explain why the pause is not an efficient way to display successive screen outputs when the output is so long that it scrolls off the screen.
4. What are the differences between the instructions PRINT TAB and PRINT SPC?
5. Why is it helpful to program our cursor control keys into a subroutine?
6. Why is it important to know how many characters are in each element of a matrix?
7. What printed message must accompany the GET or INKEY\$ statement if we are using it to halt our output displays?
8. Why is it important to use a string variable after the GET or INKEY\$ statement when we prompt "PRESS ANY KEY"?
9. What techniques are essential to the formatting of output in a user-friendly program?

Assignments:

1. Write a program that will print a 3-by-5 matrix with elements that contain up to four characters. Provide for negative entries.
2. Write a program that displays the binary, decimal, and hexadecimal equivalents of the first 75 positive whole numbers. Provide for multiple screen output.
3. Write a program that will give you the following output:

THE CURSOR IS AT COLUMN 35
4. What is the output for:

10 PRINT "THE CURSOR CAN BE FOUND AT COLUMN POSITION ";P(0)

Topic 29 SCREEN FORMATTING

Performance Objectives: Students will be able to:

- . create text screens that display information in a neat, orderly, and readable form
- . use techniques that create attractive text screens
- . program a computer to output tabular information

Vocabulary: Review: FOR/NEXT, ";", ",", LEN, TAB

Motivation:

- . Look at this NAME program and describe what it will do. (read names and print names using a FOR/NEXT loop) Then describe the format of names as they will appear on the text screen. (vertical along the left margin)

```
10 FOR I = 1 TO 10
20 READ A$
30 PRINT A$
40 NEXT I
50 END
60 DATA JOHN, SUE, MARY, BILL, HELEN
70 DATA ANTHONY, CHRIS, BARBARA, STEVE, CATHY
```

- . How would you modify this program so that the names are printed horizontally? (Use ",", or ";" at the end of statement 30.)
- . Think about how the same information can be displayed to make the output neater and more interesting.

Aim: How can we make text screens look neat, orderly, and interesting?

Development:

1. Review the use of ";" and "," in printing.
 - . Which of the two is more suited for tabular information? (",")
 - . What are some of the limitations of the function of a trailing comma in a print statement? (limited number of columns)
2. What BASIC command can be used to space printed characters? (Common to most versions of BASIC is the TAB command, which can be used to control where printing will be done along a given line.)
 - . What is the TAB command? (The form PRINT TAB(x) is a way of printing characters in a position relative to the left margin of the screen.)
3. Explain the use of the PRINT TAB(x) command. (The value of x determines the number of spaces from the left margin where printing will begin. This is much like the TAB function on an ordinary typewriter.)

- . Change line 30 in the above program to read:

```
30 PRINT TAB(20);A$
```

- . Experiment with different values of x.

- . The value of x can be a computed value. Add these lines to the above program:

```
25 K= (I * 3 + 10)
30 PRINT TAB(K); A$
```

Describe the output. (Each name is printed below and three spaces to the right of the previous one.)

4. How can we modify our name program so that we will get three columns of names in the 10th, 20th, and 30th column positions?

Acceptable answer:

```
5 K= 0
25 K=K+10
27 IF K > 30 THEN K=10 : PRINT
30 PRINT TAB(K); A$;
```

- . What is the purpose of line 5? (to initialize K)
- . What is the purpose of line 25? (to increment the value of K)
- . What is the purpose of line 27? (to recycle the value of K and begin a new line)

5. How can we center output?

Acceptable answer:

```
10 A$ = "PROGRAM TITLE"
20 K= (40 - LEN ( A$ ))/2
30 PRINT TAB(K);A$
```

Note: If your computer has 64 characters per line, change the number 40 to 64. For a 32-column computer change it to 32.

6. How can we add some interest to our text screen?

- . Add some interesting strings such as in the programs below:

Program 1:

```
10 Z$="*****"
20 A$ =" RIGHT ANSWER "
30 PRINT TAB(10);Z$
40 PRINT TAB(20);A$
50 PRINT TAB(10);Z$
```

Program 2:

```
10 X$="*           *"  
20 Y$="  *         * "  
30 Z$="      *      "  
40 A$ ="YOU'RE GREAT"  
50 PRINT TAB(10);X$  
60 PRINT TAB(10);Y$  
70 PRINT TAB(10);Z$  
80 PRINT TAB(15);A$  
90 PRINT TAB(10);Z$  
100 PRINT TAB(10);Y$  
110 PRINT TAB(10);X$
```

Applications:

1. Improve the name program so that the names are printed vertically and centered on their lines. Hint: Use a subroutine to save work.

Acceptable answer:

```
10 FOR I = 1 TO 10  
20 READ A$  
25 GOSUB 80  
30 PRINT TAB(K);A$  
40 NEXT I  
50 END  
60 DATA JOHN, SUE, MARY, BILL, HELEN  
70 DATA ANTHONY, CHRIS, BARBARA, STEVE, CATHY  
80 K= (40 - LEN ( A$ ))/2  
90 RETURN
```

Note: Remember to change the number 40 in line 80 to accommodate your computer's line length.

2. Modify the above program so that the output is surrounded by a border of asterisks:

Acceptable answer:

```
5 Z$="*****"  
  :REM NUMBER OF * EQUAL TO WIDTH OF TEXT SCREEN  
7 PRINT Z$  
30 PRINT TAB(1); " "; TAB(K); A$; TAB(40); " "  
45 PRINT Z$
```

Summary:

1. How can we output data into neat columns?
2. What method can we use to center output?
3. How can we create a border around the text screen?

Assignments:

1. Write a program that will read 12 names and print them in two columns.
2. Write a program using an INPUT statement that requests a user's name. When the name is supplied, the screen will clear and the name will be centered on its line.
3. Have a computer print a one-line message surrounded by a border of dollar signs.
4. Using asterisks inside of strings, have a computer print a diamond shape.

Assignments for More Advanced Students:

1. Create a giant letter of the alphabet using PRINT TAB statements and the text character of that letter. For example:

2. Have a computer print a giant phrase using any character you select.

Topic 30

NEAT AND ORDERLY MESSAGES

Performance Objectives: Students will be able to:

- . create and use single message displays on a computer
- . use the HOME or CLS command to emphasize output from the computer
- . use a variable loop subroutine to create a moving message

Vocabulary: New: variable loop subroutine, timing loop, HOME or CLS
Review: PRINT TAB, FOR/NEXT LOOP, variable, subroutine

Motivation:

- . When you complete the instructions below, please turn your paper face down and raise your hand to signal completion. Work as quickly as you can. Ready, set, go!

1. First read all instructions carefully.
2. Name 3 American Presidents who served after the year 1900.
3. List 5 states that have a seacoast.
4. List 7 BASIC commands.
5. Name 4 types of trees found in the area.
6. Name 3 foreign heads of government.
7. How does a computer store information?
8. Do instruction 10 only.
9. Find the product of 83 and 17.
10. Turn your paper over and raise your hand before you perform any of the above chores.

(Students who successfully follow these instructions will finish quickly because instruction 1 requires reading all the instructions. Instruction 8 says to follow only instruction 10.)

- . Give everyday examples of situations in which it is important to read instructions carefully.
- . If the instructions above were like a BASIC program, instruction 8 would be like what command in BASIC? (like a GOTO 10)
- . Instruction 10 would be like what command in BASIC? (END or STOP)

Aim: How can we program a computer to emphasize important messages?

Development:

1. How could we use the HOME or CLS command to flash an important message on the screen? (PRINT - HOME - PRINT - HOME)
2. Why doesn't the program that follows produce a satisfactory flash? (computer reprints message too quickly)

```

10 HOME
20 PRINT "PRESS THE LETTER 'Y' IMMEDIATELY"
30 GOTO 10

```

3. How can we control the timing of the flashing message? (Use a timing loop such as: FOR I = 1 TO 500: NEXT I)
4. Where would you place the timing loop for maximum effect? (Between 10 and 20 and between 20 and 30)
5. Discuss why two timing loops present a better flash than only one loop.
6. How could we use timing loops in a subroutine?

```

10 HOME
20 GOSUB 60
30 PRINT "PRESS THE LETTER 'Y' IMMEDIATELY"
40 GOSUB 60
50 GOTO 10
60 FOR I = 1 TO 500: NEXT I
70 RETURN

```

7. A variable loop subroutine is a loop in a subroutine whose limit is a variable that is defined in the main body of a program. For example:

```

10 HOME
20 X= 500 : GOSUB 60
30 PRINT "PRESS THE LETTER 'Y' IMMEDIATELY"
40 X=1000 : GOSUB 60
50 GOTO 10
60 FOR I = 1 TO X : NEXT I
70 RETURN

```

8. What advantage does a variable timing loop have? (better control of timing)
9. Let's see how we can use another variable loop subroutine to control where a message will appear on the screen. Add the lines:

```

25 L = 10: GOSUB 80
80 FOR I = 1 TO L : PRINT : NEXT I
90 RETURN

```

What would this mechanism control? (the number of blank lines to PRINT before the message.)

Application:

1. Write a program that moves an important instruction down the text screen for emphasis.

Acceptable answer:

```
10 REM MOVE INSTRUCTION DOWN
20 L=0
30 FOR J= 1 TO 10
40 HOME
50 X = 100: GOSUB 100
60 L= L + 2
65 GOSUB 120
70 PRINT TAB(10);"HELP! I'M DROWNING"
80 X = 1000: GOSUB 100
90 NEXT J
99 END
100 FOR I =1 TO L: NEXT I
110 RETURN
120 FOR I = 1 TO L: PRINT: NEXT I
130 RETURN
```

Summary:

1. Why is it important to clarify user instructions?
2. Describe the mechanism for creating a flashing message. (REPEAT: HOME-PAUSE - PRINT - PAUSE)
3. How does a variable loop subroutine work?
4. What are the advantage of a variable loop subroutine?

Assignments:

1. Write a program to flash an instruction at the bottom of the screen.
2. Write a program showing a flashing asterisk falling.
3. Write a program that will flash a message in the middle of the screen, pause, then flash a different message.
4. Describe how the HOME or CLS command is important in flashing or simulated movement routines.

Assignments for More Advanced Students:

1. Create a program that will move a flashing message up and down a text screen.
2. Create a variable loop subroutine to control a PRINT TAB statement, so that a word will slide across the text screen.

Topic 31

USING STILL GRAPHICS TO CLARIFY SCREENS

Performance Objectives: Students will be able to:

- . use still graphics to clarify and elucidate business and CAI application programs
- . implement string-defined characters to enhance screen displays

Vocabulary: New: string variable, string

Review: string, GOSUB, FOR/NEXT, timing loop, single-dimensioned array, DIM

Motivation:

- . We are going to perform a visual experiment. First type the program listed below that is compatible with your microcomputer:

```
10 REM APPLE VERSION
20 HOME : VTAB 10: HTAB 5
30 INPUT "TYPE ANY KEY ==> ";KE$
40 HOME : FOR I = 1 TO 960: PRINT KE$;:NEXT I
50 VTAB 10: HTAB 10: PRINT "PURE PROGRAMMING POWER"
60 FOR I = 1 TO 1000: NEXT I
70 VTAB 10: HTAB 10: PRINT "          " :REM 22 SPACES
80 FOR I = 1 TO 1000: NEXT I
90 GOTO 50
```

```
10 REM TRS 80 VERSION
20 CLS
30 INPUT "TYPE ANY KEY ==> ";KE$
40 CLS : FOR I = 1 TO 1024: PRINT KE$;:NEXT I
50 PRINT@20,"PURE PROGRAMMING POWER";
60 FOR I = 1 TO 1000: NEXT I
70 PRINT@20,"          " ;:REM 22 SPACES
80 FOR I = 1 TO 1000: NEXT I
90 GOTO 50
```

```
10 REM COMMODORE VERSION
20 CLS
30 INPUT "TYPE ANY KEY ==> ";KE$
40 CLS : FOR I = 1 TO 960: PRINT KE$;:NEXT I
50 VTAB1:HTAB10:"PURE PROGRAMMING POWER";
60 FOR I = 1 TO 1000: NEXT I
70 VTAB1:HTAB10:"          " ;:REM 22 SPACES
80 FOR I = 1 TO 1000: NEXT I
90 GOTO 50
```

- . Type RUN. Enter any key and press return. Observe the screen and note your observations on paper.
- . Experiment with different keys or symbols to see which are more visually appealing.

Aim: How can we use string graphics for clarity and visual effect when writing programs?

Development:

1. What is the definition of a string? (a set of characters) How are strings labeled in a computer program? (a letter or letters followed by a \$)
2. By combining previously defined strings in programs along with the ability of a computer to control where each string will be printed on the screen, many interesting effects can be created. Try the emphasized message program:

```
10 REM EMPHASIZED MESSAGE PROGRAM
20 HOME: INPUT "MESSAGE ==>"; M$
30 S$="*****"
40 HOME:PRINT S$
50 PRINT TAB(10); M$
60 PRINT S$
```

Note: Your machine may use CLS instead of HOME.

3. Strings may be individually defined and then assembled to create a variety of useful shapes. Try the following program:

```
10 REM A STRING ARROW
20 A$ = "      *  "
30 B$ = "      *  "
40 C$ = "*****"
50 D$ = "      *  "
60 E$ = "      *  "
70 HOME: GOSUB 100
80 END
100 PRINT A$:PRINT B$:PRINT C$:PRINT D$:PRINT E$
110 RETURN
```

4. Often an instruction can be emphasized by surrounding it with another symbol such as an asterisk. How would you program a computer to display the phrase "PRESS ANY KEY" inside a box made of asterisks?

Acceptable answer:

```
10 REM SURROUND A MESSAGE
20 A$ = "*****"
30 B$ = "* PRESS ANY KEY *"
40 C$ = "*****"
50 PRINT A$:PRINT B$:PRINT C$
```

- . Note that the use of string variables allows for flexibility in programming. For example, after the above program segment is executed and the user responds by pressing any key, B\$ could be redefined as "* THANK YOU *". Then by going to line 50 a new message can be PRINTed.

- . How can we create an electronic message program which will request short messages and flash them on the computer screen?

Acceptable answer:

```

10 REM ELECTRONIC MESSAGE PROGRAM
20 HOME: VTAB 10: HTAB 5
30 INPUT "HOW MANY MESSAGES ";M
40 HOME: FOR I= 1 TO M: PRINT "MESSAGE #°";I;
      "=> ";:INPUT M$(I):NEXT I
50 S$="*****"
60 FOR I = 1 TO M
70 HOME: PRINT S$
80 PRINT TAB( (40 - LEN (M$(I)))/2)
90 PRINT S$
100 FOR J = 1 TO 2000: NEXT J
110 NEXT I

```

Note: You may need CLS instead of HOME, and VTAB 10 for your machine may be ten PRINT statements.

- . What type of variable is M\$(I)? (single dimension array)
- . If more than ten messages are to be input, what must be added to the program to prevent an error? (a DIM statement)
- . What is the purpose of line 100? (timing loop)

Application:

1. One common business graphic is the bar graph. Describe in theory how you would create a bar graph with string graphics.

Summary:

1. What function(s) would still graphics serve in programming?
2. How could a shape be drawn using string-defined characters?
3. Why do programmers prefer to use strings instead of direct PRINT statements?

Assignments:

1. Using string-defined characters, write a program which will draw an outline of a tree.
2. Write a program that will draw an arrow pointing to an important message on the screen.
3. Write a program that will PRINT string-defined characters in the shape of a happy face.

Assignments for More Advanced Students:

1. Create a program that will make an asterisk arrow move across the screen.
2. Create a program that will show two lines of asterisks between which a message is centered and flashes.

Topic 32

WRITING CONCISE PROGRAMS

Performance Objectives: Students will be able to:

- . trace a program containing an ON...GOTO...statement
- . rewrite programs with multiple IF/THEN statements in shorter form with one ON...GOTO...statement
- . rewrite programs with repetitious statements using a FOR/NEXT loop
- . rewrite programs with INPUT and prompt employing INKEY\$ or GET\$ and prompt

Vocabulary: New: ON...GOTO...

Review: INT, IF/THEN, FOR/NEXT, INPUT, INKEY\$, GET\$

Motivation:

- . One of the more popular uses for a computer is in the area of visual arcade games. A popular dice game, craps, has automatic winners, automatic losers, and other rolls that serve as points and require that the player roll the dice repeatedly.
- . Elicit which sums are the winners (7, 11) and which are the losers (2, 3, 12).
- . What statements could we code to tell a user: (1) if the sum of two dice was 2, 3, or 12, and, if so, that he or she automatically lost; (2) if the sum was 7 or 11 and, if so, that he or she automatically won; and (3) if the sum was anything else, what the sum was and that he or she should roll again?

Acceptable answer:

```
100 IF SUM = 2 THEN 200
110 IF SUM = 3 THEN 200
120 IF SUM = 12 THEN 200
130 IF SUM = 7 THEN 300
140 IF SUM = 11 THEN 300
150 PRINT "Your point is " SUM ". Press 'R' to roll again."
```

Aim: How can we take a series of IF/THEN statements and rewrite them in one statement?

Development:

1. Introduce the ON...GOTO...statement:

```
50 IF INT(X) = 1 THEN 100
60 IF INT(X) = 2 THEN 200
70 IF INT(X) = 3 THEN 300
80 IF INT(X) = 4 THEN 400
```

can be rewritten as 50 ON X GOTO 100,200,300,400

Note: If $X < 1$ or $X > 4$ you might get an error message. On some

machines control will be passed on to the statement following the ON...GOTO...statement if the value of the assigning variable is not on the list. On machines that generate errors this can be circumvented by:

49 ON ERROR GOTO 500 statement

On other machines:

49 IF X<1 or X>4 THEN 500 statement

2. Review the INT function.

. If X = 3, what line would the program branch to? (300)

. If X = 2.4, what line would the program branch to? (200)

. If X = 5, what line would the program branch to? (Either there will be an error message and no branching or, in some cases, the program will go to the line after ON...GOTO...)

3. Rewrite the motivation statements with ON...GOTO...

100 ON (SUM-1) GOTO 200, 200, 150, 150, 150, 300, 150, 150, 150, 300, 200

or

100 ON SUM GOTO 200, 200, 200, 150, 150, 150, 300, 150, 150, 150, 300, 200

Here we put in any line number for the impossible sum of 1.

Applications:

1. Write these statements in more concise code using ON...GOTO...

```
60 IF R = 1 THEN 300
70 IF R = 2 THEN 400
75 IF R = 3 THEN 500
80 IF R = 4 THEN 150
81 IF R = 5 THEN 20
95 IF R = 6 THEN 900
```

Acceptable answer:

60 ON R GOTO 300, 400, 500, 150, 20, 900

2. Write a line of code that would eliminate the possibility of getting an error message from a mistyped value of R in the above example.

Acceptable answer:

59 IF R<1 OR R>6 THEN 700

3. Tell what line the program would branch to when it reaches the line:

```
130 ON Q GOTO 28, 37, 105, 240, 390, 856, 10, 571
```

when Q =

- (a) 3 (105)
- (b) 6 (856)
- (c) 5/3 (28 INT(1.66666)= 1)
- (d) 3/5 (error message INT(0.6)= 0)
- (e) 8/4 (37)

4. Write a line of code that would take the place of

```
50 IF M = 100 THEN 250
60 IF M = 200 THEN 260
70 IF M = 300 THEN 270
80 IF M = 400 THEN 280
```

Answer:

```
50 ON M/100 GOTO 250, 260, 270, 280
```

5. You are asked to help with a project for a geography class. Rewrite this program in more concise code.

```
10 PRINT "Which of these natural wonders would you like to visit?"
20 PRINT " (1) Niagara Falls"
30 PRINT " (2) Grand Canyon"
40 PRINT " (3) Yellowstone Park"
50 PRINT " (4) The Badlands"
60 PRINT
70 PRINT "TYPE IN '1', '2', '3' or '4' and press <ENTER> or <RETURN>"
80 INPUT N
90 IF N = 1 THEN PRINT "You would have to travel to New York State to see
    Niagara Falls."
100 IF N = 2 THEN PRINT "You would have to travel to Arizona to see the
    Grand Canyon."
110 IF N = 3 THEN PRINT "You would have to travel to Wyoming to see
    Yellowstone Park."
120 IF N = 4 THEN PRINT "You would have to travel to either North or
    South Dakota to see the Badlands."
130 PRINT " Follow instructions":GOTO 70
```

Acceptable answer:

The lines through 60 remain the same, and then:

```
70 PRINT "Type in a '1','2','3', or '4'"
80 N$ = INKEY$ or GET N$
81 IF N$ = "" THEN 80
85 N = VAL(N$)
86 IF N<1 OR N>4 THEN 70
90 PRINT "You would have to travel to ";
```

```

100 ON N GOTO 110, 120, 130, 140
110 PRINT "New York State to see Niagara Falls":GOTO 200
120 PRINT "Arizona to see the Grand Canyon": GOTO 200
130 PRINT "Wyoming to see Yellowstone Park":GOTO 200
140 PRINT "either North or South Dakota to see the Badlands"

```

6. Rewrite this program using a FOR/NEXT loop and READ statements to make it more concise:

```

10 PRINT "What is the Spanish word for 'house'?"
20 INPUT N$
30 IF N$ ="casa" OR N$="la casa" THEN PRINT "CORRECT!" ELSE PRINT " The
    word for 'house' is 'la casa'"
40 PRINT
50 PRINT "What is the Spanish word for parents?"
60 INPUT N$
70 IF N$ = "padres" OR N$ = "los padres" THEN PRINT "CORRECT!" ELSE PRINT
    "The word for 'parents' is 'los padres'."

```

etc.

Note: For machines that don't allow the Basic word ELSE, two statements can be used to replace line 30:

```

30 IF N$ ="casa" OR N$="la casa" THEN PRINT "CORRECT!"
32 IF N$<>"casa" AND N$<>"la casa" THEN PRINT "The word for 'house' is
    'la casa'"

```

Acceptable answer:

```

10 FOR X = 1 TO Z:REM Z = number of words to be tested
20 READ A$, B$, C$
30 PRINT "What is the Spanish word for '";A$;"'?"
40 INPUT N$
50 IF N$ = B$ OR N$ = C$ THEN PRINT "CORRECT!" ELSE PRINT "The word for
    '";A$;"' is '";C$;"'."
60 PRINT
70 NEXT X
100 DATA HOUSE, CASA, LA CASA
101 DATA PARENTS, PADRES, LOS PADRES

```

Summary:

1. How can we use ON...GOTO...to combine certain IF/THEN statements?
2. How may we shorten our programs?

Assignments:

1. Write a statement that will combine these lines of code:

```

200 IF T = 1 THEN 600

```

130

```
210 IF T = 2 THEN 610
220 IF T = 3 THEN 620
230 IF T = 4 THEN 630
240 IF T = 5 THEN 640
```

2. Write a line of code that will eliminate an error message in the previous example when values of T are too small or too large.
3. State what line the program will branch to after it executes:

```
100 ON Z GOTO 30, 250, 470, 81, 543, 449, 892, 340, 5
```

when Z =

(a) 4	(e) 5/2
(b) 8	(f) 7.4
(c) 2	(g) 10
(d) 5	(h) 3*2

4. Rewrite line 100 with the lines of code needed to replace the ON...GOTO... statement.
5. Write a program that will randomly generate numbers from 1 to 10 a thousand times and will then count the number of times each number appeared.
6. Write a program that will test the user's knowledge of state capitals.
7. Write a program that will offer multiple-choice questions about nimbus, cirrus, and stratus clouds.

Assignment for More Advanced Students:

1. Write a program that will simulate a craps game.

Topic 33
MACHINE LANGUAGE AND WALK THROUGH MEMORY
(COMPUTERS OTHER THAN APPLE)

Note: This lesson was designed for classes using machines other than Apple computers.

Performance Objectives: Students will be able to:

- . understand the difference between machine language and BASIC
- . understand how memory is addressed
- . use PEEK and POKE to manipulate machine memory
- . differentiate between a Monitor and monitor
- . use a Monitor
- . examine single units of memory as well as long ranges of memory
- . understand how computers allocate specific areas of memory for predetermined uses
- . examine the result of using a PEEK and a POKE command

Vocabulary: New: machine language, Monitor, Monitor write command, compiler, interpreter

Review: bit, byte, nybble, RAM, ROM, binary digits, hexadecimal

Motivation:

- . In what language do we communicate with our microcomputer? (BASIC)
- . What other languages does the computer understand? (FORTRAN, Pascal, COBOL, LISP, Logo, APL, Pilot, etc.)
- . Do all computers speak all these languages?
- . What must we do to our microcomputer so that we can program in these different languages?
- . Is there a universal language that all microcomputers understand?

Aim: What is machine language?

Development:

1. Make students aware of the fact that when they communicate with the computer using the BASIC language they are not speaking directly to the computer.
 - . Draw the analogy of two people speaking different languages, communicating with each other through an interpreter. Students should be aware that all digital computers operate in a similar fashion; they understand only two states of digital circuitry, on and off. Words like READ, GOTO, and DRAW mean nothing to the computer. How then do our modern computers understand the high-level languages that we program with?
 - . The commands we use are translated into binary number codes by a

compiler or interpreter that changes our codes into machine language. To see what this machine language looks like, we must use either a Monitor program (notice the capital M) or the BASIC word PEEK. The Monitor will allow us to examine and tamper with memory. Do not confuse it with the monitor (spelled with a lowercase m) that gives us a video display.

- . A Monitor is a computer program that allows us to manipulate RAM. What BASIC commands that you have used have given you direct access to the computer's memory? (PEEK, POKE, CALL, USR)
- 2. Return to the discussion of binary digits. Review the terms bit, byte, and nybble. Remind students that the smallest unit of memory in the machine is the bit. Explain that the term "bits" originally came from an abbreviation of "Binary digITS." Show students an example of a bit pattern that the computer can understand.

0 1 1 1 0 0 1 0 (which stands for the
off on on on off off on off lowercase letter r)

- . Review the number of possible combinations of zeros and ones that can appear in one 8-bit byte. Review with students the conversion of binary to hexadecimal and to decimal.
- . Review the convention of "!" before binary displays, "\$" before hexadecimal displays, and nothing to indicate decimal numbers.
- . Review the powers of 2 and develop the following table for one nybble:

bit pattern	binary	hexadecimal	decimal
off off off off	0 0 0 0	0	0
off off off on	0 0 0 1	1	1
off off on off	0 0 1 0	2	2
off off on on	0 0 1 1	3	3
off on off off	0 1 0 0	4	4
off on off on	0 1 0 1	5	5
off on on off	0 1 1 0	6	6
off on on on	0 1 1 1	7	7
on off off off	1 0 0 0	8	8
on off off on	1 0 0 1	9	9
on off on off	1 0 1 0	A	10
on off on on	1 0 1 1	B	11
on on off off	1 1 0 0	C	12
on on off on	1 1 0 1	D	13
on on on off	1 1 1 0	E	14
on on on on	1 1 1 1	F	15

- . Each binary nybble corresponds to one hexadecimal digit. The binary system is not only less cumbersome to work with; it also eliminates the problem of difficult numeration system conversions. Thus each byte is addressed by two hexadecimal digits.
- . Define kilobyte and megabyte.

- . Explain that the memory of a computer is numbered sequentially. Picture the memory as a vast array of mailboxes, each with a separate address. If you want to see what is located in any of the mailboxes you need only to access the box with that address. Look inside the box and find one byte.
 - . Tell students that the computer allocates two bytes for the number of each address. The first address would be \$0000. What would the last address be?
 - . Have students convert \$FFFF to a decimal numeral. Point out that they are beginning their count with zero; therefore there are 2 to the sixteenth power possible addresses for memory locations.
 - . Relate 64K to 65,536. Explain that this is the highest number of memory locations we can have without redesigning a computer.
 - . Return to the Monitor (or PEEK memory) to see what bytes are located at given addresses.
 - . Have students learn the technique of calling up a specific address.
 - . It is reasonable to expect the bytes to vary from computer to computer. This could enliven student participation. Introduce the method of examining ranges of memory.
3. Alert students to the dangers that can occur when the computer's memory is disturbed. They will soon learn how to manipulate memory. Before they begin it is necessary that they learn what is stored in different memory locations.
- . Remind the class that there is no way they can affect the ROM.
 - . Elicit that when they write they will be writing to RAM (Random Access Memory).
 - . Distribute a copy of the memory map of a standard machine.
 - . Point out the area of their machine that covers RAM and that which represents ROM.
 - . Discuss with students areas available for short machine language programs and areas where memory can be changed.
 - . Have students POKE a value into an area of available RAM and then PEEK that location to see that the value has truly been saved at that memory spot. Have them change the memory location to another value with POKE and then PEEK it to see that it has changed.

Applications:

1. Enter the Monitor program and examine the contents of specific memory.
2. Clear the screen and send the cursor to the upper left corner using either POKE or a Monitor command.

3. Use the Monitor move command to place the letter C in every position of the screen, or use POKE to do the same thing.

4. Convert the following numbers to hexadecimal:

11110100	Binary
10011001	Binary
780	Decimal
2565	Decimal

5. Convert the following numbers to decimal:

2000	HEX
30	HEX
10100101	BINARY
01101100	BINARY

Summary:

1. What are two benefits of writing in machine language?
2. Why do we write programs in BASIC rather than machine language?
3. What is the system Monitor?
4. Where is the system Monitor located?
5. What number systems does BASIC use for arithmetic operations?
6. How many binary digits are represented by one hex digit?
7. How many bytes are allocated for each memory address?
8. What area of memory can never be changed?
9. What command permits us to change the contents of RAM?
10. What command enables us to insert data into a machine language program?
11. If we wish to write on the text page from the monitor, what memory locations should we use?
12. How do screen codes relate to ASCII codes?

Assignments:

1. What is the last possible memory address in a machine that allocates three bytes for each address?
2. Write a BASIC statement and separately use POKES to white out the entire screen.
3. Write a machine language routine that will fill the text screen with stars.

Topic 34

MACHINE LANGUAGE AND WALK THROUGH MEMORY (APPLE COMPUTERS)

Note: This lesson was designed for teachers using Apple computers.

Performance Objectives: Students will be able to:

- . understand the difference between machine language and BASIC
- . understand how memory is addressed
- . differentiate between the Monitor and the monitor
- . enter and exit the Monitor
- . examine single units of memory as well as long ranges of memory
- . understand how computers allocate specific areas of memory for predetermined uses
- . move areas of memory for the purpose of inserting or clearing
- . use the Monitor write command to place letters on the text page
- . examine the result of using a PEEK, POKE, and CALL BASIC command

Vocabulary: New: machine language, Monitor, Monitor write command, Monitor move command, compiler

Review: bit, byte, nybble, RAM, ROM, binary digits, hexadecimal

Motivation:

- . In what language do we communicate with our microcomputer? (BASIC)
- . What other languages does the computer understand? (FORTRAN, Pascal, COBOL, LISP, Logo, APL, Pilot, etc.)
- . Do all computers speak all these languages?
- . What must we do to our microcomputer so that we can program in these different languages?
- . Is there a universal language that all microcomputers understand?
- . Does anyone know what I mean by the expression two state digital circuitry?

Aim: What is machine language?

Development:

1. Make students aware of the fact that when they communicate with the computer using the BASIC language they are not speaking directly to the computer.
 - . Draw the analogy of two people speaking different languages, communicating with each other through an interpreter. Students should be aware that all digital computers operate in a similar fashion; they understand only two states of digital circuitry, on and off. Words like READ, GOTO, and DRAW mean nothing to the computer. How then do our modern computers understand the high-level languages that we program with?

- . The commands we use are translated into binary number codes by a compiler that changes our codes into machine language. To see what this machine language looks like, we must enter the system Monitor (notice the capital M). This Monitor is part of ROM (Read Only Memory). Do not confuse it with the monitor (spelled with a lowercase m) that gives us a video display.
 - . The Monitor can be thought of as another computer language -- a special language that permits the programmer to deal directly with the memory of the computer. What BASIC commands that you have used have given you access to the computer's memory? (PEEK, POKE, CALL, USR)
2. Have students enter the Monitor with CALL -151. Elicit the appearance of the Monitor prompt. Discuss the "*", "]", and ">". Have students press Return a few times and describe what they see. Ask them if they see any binary digits. Let them conjecture on the meaning of the display.
- . Teach three ways of returning to BASIC. Use Ctrl-C to return to BASIC without destroying any BASIC programs that might be in memory. Use Ctrl-B to return to BASIC and clear the memory at the same time. In the event that a disk drive has been booted and DOS is in effect, students can use 3DOG. In a more advanced class the teacher could return to this command for a discussion of the Monitor Go command and its relation to the CALL.
3. Return to the discussion of binary digits. Review the terms bit, byte, and nybble. Remind students that the smallest unit of memory in the machine is the bit. Explain that the term "bits" originally came from an abbreviation of "Binary digITS." Show students an example of a bit pattern that the computer can understand.

```

0 1 1 1 0 0 1 0
off on on on off off on off

```

- . Review the number of possible combinations of zeros and ones that can appear in one 8-bit byte. Review with students the conversion of binary to hexadecimal and to decimal.
- . Review the convention of "!" before binary displays, "\$" before hexadecimal displays, and nothing to indicate decimal numbers.
- . Review the powers of 2 and develop the following table for one nybble:

bit pattern	binary	hexadecimal	decimal
off off off off	0 0 0 0	0	0
off off off on	0 0 0 1	1	1
off off on off	0 0 1 0	2	2
off off on on	0 0 1 1	3	3
off on off off	0 1 0 0	4	4
off on off on	0 1 0 1	5	5
off on on off	0 1 1 0	6	6
off on on on	0 1 1 1	7	7

on off off off	1 0 0 0	8	8
on off off on	1 0 0 1	9	9
on off on off	1 0 1 0	A	10
on off on on	1 0 1 1	B	11
on on off off	1 1 0 0	C	12
on on off on	1 1 0 1	D	13
on on on off	1 1 1 0	E	14
on on on on	1 1 1 1	F	15

- . Each binary nybble corresponds to one hexadecimal digit. The binary system is not only less cumbersome to work with; it also eliminates the problem of difficult numeration system conversions. Thus, each byte is addressed by two hexadecimal digits.
 - . Define kilobyte and megabyte.
4. Explain that the memory of the computer is numbered sequentially. Picture the memory as a vast array of mailboxes, each with a separate address. If you want to see what is located in any of the mailboxes you need only to access the box with that address. Look inside the box and find one byte.
- . Tell students that the computer allocates two bytes for the number of each address. The first address would be \$0000. What would the last address be?
 - . Have students convert \$FFFF to a decimal numeral. Point out that they are beginning their count with zero; therefore, there are 2 to the sixteenth power possible addresses for memory locations.
 - . Relate 64K to 65,536. Explain that this is the highest number of memory locations we can have without redesigning our computer.
5. Return to the Monitor to see what bytes are located at given addresses.
- . Have students learn the technique of calling up a specific address.

Call -151	See *
Type 300 <CR>	See 0300- 48
Type 100 <CR>	See 0100- 37
Type 21FF <CR>	See 21FF- 00
Type 0785 <CR>	See 0785- B5

- . It is reasonable to expect the bytes to vary from computer to computer. This could enliven student participation. Introduce the method of examining ranges of memory.
- . Have students clear the screen with "ESC@" and type 200.230 <CR>. Students should see:

```

0200- B2 B0 AE B2 B3 B0 8D
0208- 31 00 20 20 20 20 20
0210- 20 20 20 20 20 20 20
0218- 20 20 20 20 20 20 20

```

0220- 20 20 20 20 20 20 20
0228- BE A0 A0 A0 A0 A0 A0
0230- A0

- . Show students that the "\$" was not used. Explain that this is because the computer speaks only in hex. It is not necessary to differentiate between different systems of enumeration.
 - . Tell students to establish the following rule for themselves: When communicating with the Monitor, use hex; when writing in BASIC, use decimal.
6. Alert students to the dangers that can occur when the computer's memory is disturbed. They will soon learn the Monitor Write command. Before they begin it is necessary that they learn what is stored in different memory locations.
- . Remind the class that they cannot enter data into the ROM.
 - . Elicit that when they write they will be writing to RAM (Random Access Memory).
 - . Distribute a copy of the memory map of a standard 48K machine.
 - . Tell students that they will have to consult the manual for any specific computer. Point out that \$0-\$BFFF is RAM. \$C000-\$FFFF is ROM.

0.\$1FF	program work space not available to user
200.2FF	keyboard character buffer
300.3FF	available to user for short machine language programs
400.7FF	page 1 of text and color graphics
800.1FFF	basic interpreter
2000.3FFF	hi-res page 1
4000.5FFF	hi-res page 2
6000.95FF	unused
9600.BFFF	DOS
C000.CFFF	hardware I/O addresses
D000.DFFF	future ROM expansion
E000.F7FF	reserved
F800.FFFF	system monitor

- . Discuss with students areas available for short machine language programs and areas where memory can be changed.
7. Teach the Monitor Write command.": ". Enter the monitor and choose an area where we can change the memory without causing any damage. \$0400 would be a good choice.
- . First see what is in that memory location by typing 400 and then <CR>.
 - . Change the memory by typing 400.EF <CR>.
 - . Next examine the contents of 400 by typing 400 <CR> to prove that the memory has in fact been changed.

- . After several tries ask students what they think would happen if they attempted to write to ROM. Choose an address in ROM and attempt to change the memory.
 - . Check to see if the memory has been changed and discuss with students the results. Stress the fact that ROM cannot be changed. Show that it is still possible to change areas of RAM that could give us a good deal of trouble.
8. Ask students if they think it is possible to write on the text page directly from the Monitor. Elicit the address of page 1 of text.
- . Ask students if they think that printing a C to address \$610 would cause a "C" to be printed on the screen. If there is uncertainty, let them try it. The result should be an inverse L in the 5th line, 17th column.
 - . Elicit the ASCII Code system. Explain the procedure of adding 128 to the ASCII code and converting to hex. Have students type:
- 650:C1 C2 C3 C4 C5 C6 C7 <CR>
- The result will be ABCDEFG printed near the bottom of the screen.
- Then have students try:
- 680:0A 0B 0C 0D 0E 0F
- The result will be inverse JKLMNO printed on about the eighth line.
9. It can be pointed out at this time that the screen lines are not numbered sequentially. If time permits, initiate a discussion of the problems that this causes when creating graphics displays.
10. The Monitor Move command is a very useful one. It permits us to move large areas of memory to create free space that we might need for our programs. 3A0<300.341M is the format of the Monitor Move command. The symbols "<" and "M" constitute the command.
- . This statement will move the contents of the memory that is located beginning in 300 and ending in 341 to the area of memory that begins at 3A0.
 - . We do not indicate an ending number for our destination range. This procedure comes in very handy after a long machine language program has been entered and we find we omitted several sets of data.
 - . A useful routine that will impress students with the speed at which machine language programs are executed is to fill the text screen with some character. Enter an inverse "O" in \$401. Then execute the following Monitor Move command:

402<401.7FFM

Since this command operates on one byte at a time and works in ascending order, the first step will copy the inverse 0 into \$402. Notice the next register to be copied is \$402, but we have just placed the inverse 0 in there. In this manner we fill the entire range with our inverse 0.

Application:

1. Enter the monitor and complete the following tasks:

- . Examine the contents of locations \$6800 through \$6B1A.
- . Examine the contents of \$ABCD. Do not let the contents of any other register appear on the screen.
- . Clear the screen and send the cursor to the upper left corner.
- . Examine the contents of the four nonsequential registers \$30, \$5A, \$BB, and \$F0 with a command that only uses <CR> once.
- . Use the Monitor Move command to place the letter C in every position of the text screen.
- . Describe the effects of executing the following sequence of commands.

```
5FF:C1
600<5FF.7FF
```

- . What does this demonstrate about the text screen lines?
- . Return to BASIC and execute the following immediate commands:

```
POKE 768,169
POKE 769,19
POKE 770,141
PRINT PEEK(769)
```

Explain the results of the screen display. Return to the Monitor and check to see if these commands had an effect. What memory locations must be examined? What keys must be typed?

- . What are the digits that we hope to see? Make a general statement about numbers entered in BASIC and numbers entered in the Monitor.
- . Calculate the machine language code for a blank. Incorporate this with the Monitor Move command to clear the screen.
- . Convert the following numbers to hexadecimal:

```
!11110100
!10011001
780
2565
```

. Convert the following numbers to decimal:

\$2000
\$30A
!10100101
!01101100

Summary:

1. What are two benefits of writing in machine language?
2. Why do we write programs in BASIC rather than machine language?
3. What is the system Monitor?
4. Where is the system Monitor located?
5. How do we enter the Monitor?
6. What control characters are used to exit the Monitor?
7. When can we use 3DOG?
8. What kind of numbers does BASIC use?
9. What kind of numbers does the computer understand?
10. How many binary digits are represented by one hex digit?
11. How many bytes are allocated for each memory address?
12. What area of memory can never be changed?
13. What command permits us to change the contents of RAM?
14. What command enables us to insert data into a machine language program?
15. If we wish to write on the text page from the monitor, what memory locations should we use?
16. How do screen codes relate to ASCII codes?
17. Describe two ways that you can examine the contents of \$E8?

Assignments:

1. How many memory locations would be available to a system that allocated three bytes for each address?
2. Examine the contents of \$2000 to \$3FFF using a single Monitor command. Fill this section of the computer's memory with zeros. Use the method of storing \$00 in only one register and then use the Monitor Move command.

3. Write a BASIC statement that will have the same effect as the following machine language routine:

*E8:A0 03

4. Translate the following BASIC program into one that can be written to the Monitor. Carefully describe each step in the procedure that you would use to enter this program directly into the Monitor.

```
10 POKE 232,72: POKE 233,3
20 FOR LOC = 840 TO 867
30 READ V
40 POKE LOC,V
50 NEXT LOC
60 DATA 2,0,6,0,15,0
70 DATA 63,63,36,36,45,45,54,54,0
80 DATA 63,63,63,63,12,12,12,12,21,21,21,21,0
90 END
```

5. Write a machine language routine that will fill the text screen with stars.
6. Enter the Monitor and write your name and a short message on the text screen. The hex code for a carriage return is \$8D.

Topic 35

DEVELOPING MOVING GRAPHICS USING THE PRINT@ STATEMENT

Note: This lesson is designed for TRS-80 users only.

Performance Objectives: Students will be able to:

- . state the function of the PRINT@ statement
- . explain the format for the use of the PRINT@ statement
- . list the position range for each of the 16 lines on a TRS-80 screen
- . position information on the screen using the PRINT@ statement

Vocabulary: New: PRINT@

Motivation:

- . How could we get the message "Hi there, friend!" to appear in the middle of the screen?

Acceptable answer:

```
10 FOR X = 1 TO 8           :REM  skipping lines routine
20 PRINT
30 NEXT X
40 PRINT TAB(24)"Hi there, friend"
50 GOTO 50
```

- . Elicit from the pupils that there are 16 lines on a TRS-80 screen, and so we would have to drop down 8 or 9 lines to get to the center.
- . Elicit from the pupils that there are 64 columns across on a TRS-80 Model III or IV. In order to center, TAB (32 -INT(length/2)).

Note: If you have taught the LEN function, the machine can do the counting for you.

```
35 LET L = LEN("Hi there, friend!")
40 PRINT TAB( 32-INT(L/2));"Hi there, friend!"
```

- . Explain that there is a faster way we can position anything anywhere on the screen's 1024 positions. That function is called PRINT@, "Print at."

Aim: How do we position information anywhere on a TRS-80 screen using PRINT@ statements?

Development:

1. Have the students sketch a CRT with 64 columns and 16 rows. Explain the positions on the top line are numbered with integers from 0 to 63, those on the second line are 64 to 127, etc. A 64 times table will give you the starting positions for each line. Position 472 is the approximate middle. Position 0 is at the top left, position 63 top right, position 960 bottom left, position 1023 bottom right.

2. The PRINT@ format is:

line number PRINT@ <position number from 0 to 1023>, <information to be printed >

The position number cannot be negative. If a decimal fraction is entered, the machine takes the INTeGer portion of the entry and excludes the fractional part.

3. Rewrite the answer to the Motivation problem using PRINT@.

Acceptable answer:

```
10 LET A$= "Hi there, friend!"
20 LET L = LEN(A$)
30 PRINT@ 472-INT(L/2), A$
40 GOTO 40
```

Applications:

1. Write a program so that "Peek-a-boo" appears at a random position on the screen, then disappears, then reappears at another random position on the screen.

Acceptable answer:

```
10 LET A$ = "Peek-a-boo"
20 LET L = LEN(A$)
30 PRINT@ RND(1024-L)-1), A$ :REM generates from 0 to the last
   position minus size of message
40 FOR X = 1 TO 200 : NEXT X : REM time delay to see message
50 CLS
60 FOR X = 1 TO 200: NEXT X : REM time delay with blank screen
70 GOTO 30
```

2. Write a program that will take the message of your choice from the top left of the screen to the bottom right of your screen.

Acceptable answer:

```
10 LET A$= "   your message"
20 LET L = LEN(A$)
30 FOR P = 0 TO (1023 - L)
40 PRINT@ P, A$
50 NEXT P
60 END
```

3. Write a program that will take the message of your choice and have it run from the left center to the right center of the screen.

Acceptable answer:

```
10 LET A$ = " your message"
20 LET L = LEN(A$)
30 FOR P = 448 TO 511 - L
40 PRINT@ P,A$
50 NEXT P
69 END
```

4. Write a program that would blink NORTH in top center, SOUTH in bottom center, EAST in center right, and WEST in center left of the CRT.

Acceptable answer:

```
10 PRINT@ 30,"NORTH"
20 GOSUB 200
30 PRINT @ 507, "EAST"
40 GOSUB 200
50 PRINT@ 989,"SOUTH"
60 GOSUB 200
70 PRINT@ 448, "WEST"
80 GOSUB 200
90 GOTO 10
200 FOR X = 1 TO 1000 : NEXT X
210 CLS
220 FOR X = 1 TO 1000 : NEXT X
230 RETURN
```

5. Have the students create faces or stick drawings that move on the screen.

Acceptable answer:

```
10 LET A$ = "      #####"
20 LET B$ = "      X      X"
30 LET C$ = "    (X @ @ X)"
40 LET D$ = "      X  U  X"
50 LET E$ = "      X  ---  X"
60 LET F$ = "      X      X"
70 LET G$ = "      XXXXX"
75 FOR N = 64 TO 110 :REM Experiment with changing this;
include a STEP 65 !
80 PRINT@ N, A$
90 PRINT@ N+64, B$
100 PRINT@ N+ 2*64, C$
110 PRINT@ N + 3*64, D$
120 PRINT@ N + 4*64, E$
130 PRINT@ N + 5*64, F$
140 PRINT@ N + 6*62, G$
150 FOR Z = 1 TO 50: NEXT Z
160 NEXT N
```


Summary:

1. What does the PRINT@ function do?
2. What is the format for the PRINT@ function?
3. How can we determine the starting position number for line 5? $(5 - 1) * 64$
4. How can we determine the starting position number for any line from 1 to 16? $(N - 1) * 64$

Assignments:

1. Using PRINT@, write a routine that will draw a line across the tenth row of the CRT.
2. Using PRINT@, write a routine that will draw two columns of X's, in columns 20 and 40.
3. Write a program that will print "Twinkle, twinkle, little star" centered in the second line and then have "*" appear randomly, filling up the screen.
4. Write a program that will have the message of your choice move from the top left toward the bottom right of your CRT.

Assignment for More Advanced Students:

1. Write a program that will first flash "Where are you?" in the middle of the screen and then "I'm over here!" in a random position on the screen. The "Where are you?" should next flash where the "I'm over here!" just appeared. Then "I'm over here!" should flash at another random position on the screen.

Topic 36

MOVING SCREEN PATTERNS USING HTAB AND VTAB

Note: This lesson was designed for teachers using machines that allow the use of the HTAB and VTAB statements. Examples were designed to work on Apple computers but will work with slight modifications on any machines that allow the HTAB and VTAB statements.

Performance Objectives: Students will be able to:

- . use VTAB and HTAB commands to create moving objects
- . use variables with the VTAB and HTAB commands
- . understand the redrawing technique used to simulate movement
- . create programs that can move defined string variables up, down, or diagonally across the screen

Vocabulary: New: VTAB, HTAB, concatenation

Review: FOR/NEXT, string variable, timing loop

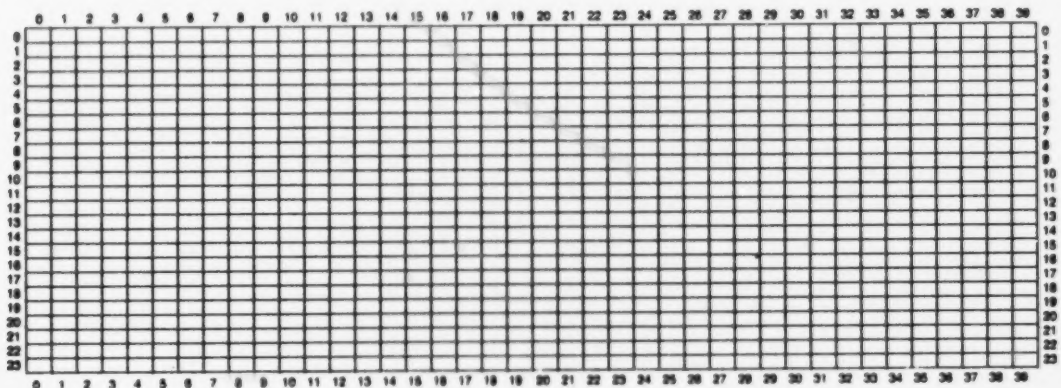
Motivation:

- . We have all seen signs that seem to move an arrow made of lights to attract attention. Upon closer inspection we realize that the arrow is made of fixed light bulbs. How would you explain the moving arrow effect?

Aim: How can we use the VTAB and HTAB commands to simulate moving graphics?

Development:

1. How does real movement differ from a computer's simulated movement? (Real movement involves a continuous physical motion that covers every point from start to finish. Simulated movement is the result of the computer's ability to draw, erase, and redraw an image so quickly that it looks like real movement.)
2. The text screen of the Apple II computer has 40 horizontal tab positions and 24 vertical tab positions. Thus we can control output to the screen by specifying a vertical and a horizontal position.



3. How could we print the letter X in the center of the text screen?
(VTAB 12: HTAB 20: PRINT "X")
4. How can we write a program that will move an asterisk from left to right, passing through the center of the screen?

Acceptable answer:

- ```

10 REM MOVE STAR LEFT TO RIGHT
20 HOME
30 FOR X = 1 TO 39 : REM generates the numbers from 1 to 40 for the
 HTAB
40 VTAB 10 : HTAB X : PRINT "*": REM specifies where to print the "*"
50 FOR I= 1 TO 500: NEXT I: REM is a timing loop
60 VTAB 10: HTAB X: PRINT " ": REM erases what we have drawn in order
 to create the illusion of movement
70 NEXT X: REM continues the cycle until the "*" is across the screen

```
5. How could we make our "\*" move across the screen from right to left?  
(change line: 30 FOR X= 39 TO 1 STEP -1)
  6. Note that the number 39 is used in our moving star program because an automatic carriage return is generated when we print in the fortieth column. The same problem occurs when printing on the twenty-fourth vertical line.

#### Application:

1. Enter the following program, which moves an arrow across the screen, and then follow steps (a) and (b).

```

10 REM SHOOT AN ARROW
20 A$ = ">>>---->" : REM the arrow is defined as a string
30 B$= " " : REM B$ is equal to two blank spaces
40 VTAB 10 : REM defines the vertical line
50 FOR X = 1 TO 25 : REM generates the numbers for the HTAB in line 60
60 HTAB X
70 FOR I = 1 TO 500 : NEXT I : REM is a timing loop
80 PRINT B$ + A$; : REM combines the two spaces and the arrow and PRINTs
 the total image.
90 NEXT X

```

Note: Without the blank spaces to the left of the arrow a trail would be left across the screen. This merging of two strings is called concatenation.

- (a) Modify the arrow program so that A\$ is defined as your first name. See your name slide across the screen.
- (b) How could you write a program which will slide your name continuously back and forth across the screen?

Acceptable answer:

```
10 REM NAME SLIDE
20 A$ = "YOUR NAME"
30 B$= " "
40 VTAB 10
50 FOR X = 1 TO 25
60 HTAB X
70 FOR I = 1 TO 150 : NEXT I
80 PRINT B$ + A$;
90 NEXT X
60 FOR X = 29 TO 1 STEP -1
70 HTAB X
80 FOR I = 1 TO 150 : NEXT I
90 PRINT B$ + A$;
100 NEXT X
110 GOTO 50
```

Summary:

1. What are the numerical limits of VTAB and HTAB?
2. Why are variables better to use with the VTAB and HTAB method of moving graphics? (fewer programming statements, easier to modify, more efficient)
3. What is the difference between real movement and the computer's simulated movement?
4. How can we simulate movement with the VTAB and HTAB commands?

Assignments:

1. Modify the arrow program so that it will leave a trail as it moves across the screen.
2. Write a program that makes your name move across the screen slowly at first and then faster and faster.
3. Modify the arrow program so that the arrow will move from left to right, then turn around and point to the left, then move from right to left.
4. Write a program to move the "\*" diagonally from the upper left corner to the lower right corner.

Assignments for More Advanced Students:

1. Write a program that will scroll your name from top to bottom and then from bottom to top.
2. Write a program that will move your name around the border of the text screen.

### Topic 37

## MOVING GRAPHICS TECHNIQUES USING THE PRINT@, SET, AND RESET

Note: This lesson is designed for TRS-80 model III AND IV users only.

Performance Objectives: Students will be able to:

- . recall uses and format of PRINT@
- . "turn on" small blocks of light on the screen using the SET function
- . "turn off" small blocks of light using the RESET function
- . create moving graphics using the SET, RESET, and PRINT@ functions

Vocabulary: New: SET, RESET, pixel  
Review: PRINT@

Motivation:

- . How can we make large initials out of individual letters and have them flash near the center of the CRT? An example of such a large initial is:

```
FFFFFFFFF
F
FFFFF
F
F
F
```

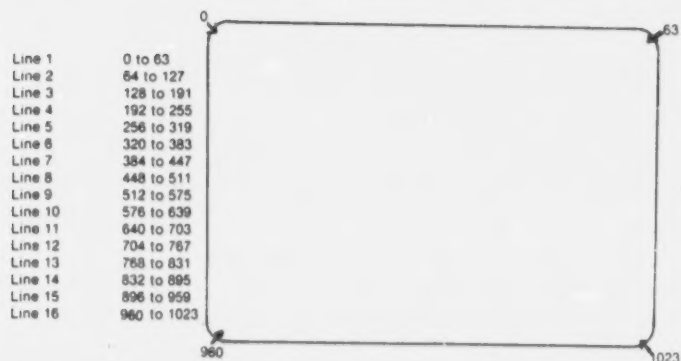
Acceptable answer:

```
10 PRINT@ 467, "FFFFFFFFF"
20 PRINT@ 467+ 64, "F"
30 PRINT@ 467+ 64*2, "FFFFF"
40 PRINT@ 467+ 64*3, "F"
50 PRINT@ 467+ 64*4, "F"
60 PRINT@ 467+ 64*5, "F"
70 FOR X = 1 TO 150: NEXT X
80 CLS
90 FOR X = 1 TO 150 : NEXT X
100 GOTO 10
```

Aim: How can we code programs to give us movable graphics?

Development:

1. Review the 1024 screen positions and the PRINT@ positions running from 0 to 63 horizontally with 16 lines vertically. (See the figure at the top of the next page.)
2. Explain that there is a SET function that divides the screen into smaller units: 128 horizontally (from 0 to 127) and 48 vertically (from 0 at the top to 47 at the bottom). It allows selective blocks of light called pixels to be lit. The word pixel comes from "picture cell."



3. This is the format for the SET function:

FORMAT: line number SET(h,v)

where h equals the number of columns from the left edge and v equals the number of rows down from the top of the screen.

For example:

10 SET(18, 27)

will light up a little square of light 18 columns from the left edge and 27 rows down from the top of the screen.

#### Applications:

1. Draw a horizontal line five units down from the top, using SET.

Acceptable answer:

```
10 FOR CO = 0 TO 127
20 SET(CO, 5)
30 NEXT CO
```

2. Draw a vertical line down the center of the screen using SET.

Acceptable answer:

```
10 FOR RO = 1 TO 47
20 SET(64, RO)
30 NEXT RO
```

3. Draw a rectangle on the screen from coordinates (5,6) to (122,6) to (122,41) to (5,41) and back to (5,6).

Acceptable answer:

```
5 REM *** Horizontal line routine ***
10 FOR CO = 5 TO 122
20 SET(CO,6)
30 NEXT CO
```



```

40 REM *** Vertical line routine ***
50 FOR RO = 6 TO 41
60 SET(122,RO)
70 NEXT RO
80 REM *** Horizontal line from right to left ***
90 FOR CO = 122 TO 5 STEP -1
100 SET(CO,41)
110 NEXT CO
120 REM *** Vertical line from bottom to top ***
130 FOR RO = 41 TO 6 STEP -1
140 SET(5,RO)
150 NEXT RO
200 GOTO 200 :REM this keeps image on screen

```

4. Insert a message inside the border you programmed for Application 3.

Add:

```

160 PRINT@ 404, "SAM"
170 PRINT@ 532, "Loves" :REM two lines down
180 PRINT@ 660, "LANEY"

```

5. "Paint" the whole screen white.

Acceptable answer:

```

10 FOR CO= 0 TO 127
20 FOR RO = 0 TO 47
30 SET(CO,RO)
40 NEXT RO
50 NEXT CO

```

6. "Punch" holes in the white screen created in Application 5 by turning off 100 pixels.

Add:

```

60 FOR X = 1 TO 100
70 LET CO = RND(128)-1
80 LET RO = RND(48)-1
90 RESET(CO,RO)
100 NEXT X

```

7. Write a program that will send a blip of light across the screen.

Acceptable answer:

```

10 FOR CO = 1 TO 127
20 SET(CO,24)
30 RESET(CO-1,24)
40 FOR Q = 1 TO 10 : NEXT Q
50 NEXT CO

```

8. Modify the program you wrote for Application 7 to put a target at

position (100,24). When the blip reaches the target, have both the blip and the target disappear and have a message appear at the center saying that the target was hit.

Summary:

1. How can multiline graphics be coded into BASIC? (Assign the position and add a multiple of 64 to that position for subsequent lines.)
2. How do we get graphic representations to move with PRINT@? (Use a FOR/NEXT loop)

Assignments:

1. Simulate a computer and tell what would happen during the execution of this program:

```
10 REM *** Create a picture ***
20 PRINT@ 380, "TYPE 'D' FOR DOWN, 'U' FOR UP"
30 PRINT@ 504, "TYPE 'R' FOR RIGHT, 'L' FOR LEFT"
40 SET(CO,RO)
50 IS= INKEY$
60 IF IS = "" THEN 50
70 IF IS = "D" THEN RO = RO + 1
80 IF IS = "U" THEN RO = RO - 1
90 IF IS = "R" THEN CO = CO + 1
100 IF IS = "L" THEN CO = CO - 1
110 GOTO 40
```

2. Modify the program in Assignment 1 so that if the letter E were pressed the machine would enter a routine that would turn off the set points until C (for create) was pressed. The program would create a picture again.
3. Write a program that would turn on lights randomly all over the screen.
4. Write a program that would draw a rectangle and then turn on lights randomly within that figure. You can make the rectangle look like a window with stars appearing inside.
5. Paint the screen white and then draw a black rectangle in it.
6. Write programs that will draw letters 8 to 10 rows high using SET. Move the letters across the screen from left to right.
7. Write a program that will draw a small rectangle inside a larger rectangle.

Assignment for More Advanced Students:

1. Change the program from Application 8 to create an explosion with blips flying in all directions.

## **Topic 38**

### **MOVING BLOCK GRAPHICS WITH THE HTAB AND VTAB**

Note: This lesson was designed for machines that allow the use of the HTAB and VTAB statements.

Performance Objectives: Students will be able to:

- . develop simulated moving block graphics routines using VTAB and HTAB commands
- . use the GET statement to control object movement
- . use the IF/THEN statement to control the limits of object movement

Vocabulary: New: GET, CHR\$, VTAB, HTAB

Review: INVERSE, GET, NORMAL, VTAB, HTAB, subroutine, IF/THEN

Motivation:

- . Moving objects on the display screen can be controlled with either joysticks, paddles, or keyboard entry.
- . Draw a number line from 1 to 40. Place a marker over the number 20. Then flip a coin to determine which direction to move the coin. Heads will move the coin to the right, and tails will move it to the left. Do this for approximately 20 tosses of the coin, and after each toss record the number the marker is on.

This exercise illustrates how the Apple II computer can control where a character will be placed. Each time the coin came up heads the position increased by one. This position can be stored in a variable called H and can be used to keep track of our object, as well as to limit its movements.

Aim: How can we use key input to control the movement of a string-defined object using VTAB and HTAB commands?

Development:

1. If H is the variable that controls the horizontal placement of a character, how can you prevent the object from going past the right or left margins of the screen?

Acceptable answer:

```
IF H > 40 THEN H = 40
IF H < 1 THEN H = 1
```

2. Our objective is to write a program that draws a cannon on the screen; when we type the right arrow it should move to the right, and when we type the left arrow it should move left. Enter the following program into the computer:

```
1 REM CANNON MOVE PROGRAM
5 HOME
10 A$ = " ":B$ = " ":C$ = " " : REM Defines three strings that
 will be the building blocks of our cannon
```

```

15 V = 10:H = 20 : REM Initializes our vertical and horizontal variables
20 HOME : GOSUB 200 : REM Clears screen and calls the print cannon
 subroutine
30 GET I$: REM waits for keyboard input
40 IF I$ = CHR$(8) THEN H = H - 1 : REM Decreases value of H when left
 arrow is pressed
50 IF I$ = CHR$(21) THEN H = H + 1 : REM Increments the value of H when
 the right arrow is pressed
60 GOTO 20 : REM Continues the loop
200 REM Lines 200-260 Are the subroutine that prints our cannon.
210 INVERSE
220 VTAB V: HTAB H: PRINT A$
230 VTAB V + 1: HTAB H - 1: PRINT B$
240 VTAB V + 2: HTAB H - 2: PRINT C$
245 VTAB V - 1: HTAB H : REM Locates the cursor for the top of our
 cannon.
250 NORMAL
260 RETURN

```

#### Applications:

1. How can we use other keys besides the left and right arrow to move the cannon? (Use the GET I\$ then IF I\$ = "A" THEN ...or IF I\$ = "I" THEN ...)
2. How can we modify the "cannon move" program so that the cannon will not scroll past the left or right margin?

#### Add:

```

215 IF H < 5 THEN H = 5
217 IF H > 38 THEN H = 38

```

3. How can we create something other than a cannon? (Use other definitions for the string variables. Example: A\$= "X":B\$="XXX":C\$="XXXXX")

#### Summary:

1. How can we control and keep track of our moving objects? (by use of variables which change according to conditions such as key entry)
2. Why are subroutines useful for drawing objects? (fewer programming statements, easier to modify, and more efficient)
3. How can we change the shape of the object shown on the screen? (Redefine the variable used to print the object.)

#### Assignments:

1. Create a program that will move your name left when you type the L key and right when you type the R key.
2. Create a program that moves an asterisk up the screen when the U key is typed and down when the D key is typed.

3. Modify the above program so that the asterisk can also move left when the L key is pressed and right when the R key is pressed.
4. Write a program that moves a character around the screen at random. When the space bar (CHR\$(32)) is pressed it will freeze that object on the screen and a new shape will appear at random, continuing the cycle until six characters are frozen on the screen.

Assignments for More Advanced Students:

1. Modify the cannon program so that when the space bar is pressed a bullet (use ".") is fired and travels up the screen. (Hint: Use the space bar to launch the bullet. The space bar is CHR\$(32).)
2. Enhance the above program by adding a target at the top of the screen, and when the missile is at the same location as the target create a simulated explosion.

## *Topic 39*

### RAM, ROM, AND OTHER MEMORY

Performance Objectives: Students will be able to:

- . name two types of storage chips
- . explain the function of a microprocessor
- . list four types of microprocessors
- . explain what an address bus is
- . describe how a computer's memory is organized
- . explain what video RAM is
- . explain what pixels are and how they are turned on
- . explain what PEEK and POKE statements are

Vocabulary: New: ASCII code, pixel, PEEK, POKE

Review: integrated circuit, LSI, VLSI, RAM, microprocessor

Motivation:

- . Bob wrote a mailing label program for his computer club. The name, street address, city, state, and zip code of each member can be entered into the computer and the program can then print a mailing label for the club newsletter. Bob entered his program and spent an hour INPUTING all the data. He printed his first set of labels and then saved his program on a disk. At the time of the next mailing he loaded the program from the disk, ran it, and found that although the program had been saved none of the data was present. Why?

Aim: How is the memory of a computer organized?

Development:

1. Briefly review the history of the computer touching on the following topics:

- . Abacus and Pascal's calculator - concept of place value.
- . MARK I - electromechanical computer.
- . ENIAC - vacuum tube technology. The tubes act like the gears in Pascal's calculator or like the electromechanical switches in the MARK I; they do the counting.

The greater the number of vacuum tubes in a computer the greater the number of operations that can be performed. Vacuum tubes generate a lot of heat, burn out often, and take up a lot of room.

- . UNIVAC - vacuum tube technology.
- Stored program concept: Explain that until the UNIVAC the data to be worked on was entered into the computer while the instructions for manipulating that data (the program) were wired into the computer. Changing the program meant rewiring the circuitry in early models or unplugging and plugging wires in later models.



- In the UNIVAC both the data and the program were entered into a computer.
- . Transistors - Transistors are made of impure silicon chips that conduct small amounts of electricity. Transistors generate less heat and burn out far less frequently than do vacuum tubes. Furthermore, since they are smaller, more of them can occupy a given space than can vacuum tubes. The more transistors (switches), the more calculations that can be done or the faster they can be done or the greater the amount of data that can be stored in the computer (the larger its memory).
- . Integrated circuitry - Transistors are silicon chips with wires attached to them, packaged in a small container and attached to other transistors by wires. To reduce the space required, integrated circuits were invented. To make an integrated circuit, impurities are placed at different locations on a single silicon chip, and these locations are connected in the desired order by very fine "wires." Thus in a very small area the equivalent of approximately 60 transistors and other electrical components can be stored.



Again, the speed of calculating increased, computing power increased, and reliability increased.

- . LSI - Large Scale Integration. The equivalent of tens of thousands of transistors were placed on a chip 1 centimeter on each side.

It was the development of LSI technology that enabled the development of microcomputers.

- . VLSI - Very Large Scale Integration. The equivalent of hundreds of thousands (400,000 and more) of transistors were placed on a chip 1 centimeter on each side.

It is the continuing advances made in this field, in which ever-increasing numbers of transistors are placed on a 1 centimeter square chip (1 million at last count) that enable us to constantly increase the power of microcomputers while reducing their size.

2. Distinguish between the following types of chips found in the microcomputer. They are similar in that they are all made with the same technology. They differ in that their wiring is different so that they all do different things.

- . There are two kinds of storage chips: RAM and ROM.

- RAM (Random Access Memory) is used for temporary storage of information (the "blackboard" of the computer). All programs and data are stored on RAM chips, and they can be erased either by program commands or by turning off the computer.
- ROM (Read Only Memory) contents are wired in at the factory and the user can only use or copy the contents but cannot change the contents in any way. ROM information is not lost when the computer is turned off. ROM contains the operating system of the computer and usually contains its BASIC.

Note: The existence of PROM and EPROM chips can be mentioned at the instructor's discretion.

2. The microprocessor is the chip that does the calculating and/or manipulation of data.
  - Indicate that processors differ in the number of bits they can manipulate at the same time. Relate that this makes a difference in the speed at which the processor accomplishes its work.
  - List the following types:
    - 4-bit - used in calculators and watches
    - 8-bit - used in most current computers
    - 16-bit - used in the newer generation of computers
    - 32-bit - excellent for doing rapid calculations; used in microcomputers doing graphics and design work
  - Indicate that the amount of memory directly addressable by the microprocessor is limited by its design.
  - Relate that all memory locations in both the RAM and ROM chips of a computer are sequentially numbered from 0 up to 65536 in a 64K computer. Indicate that when the microprocessor needs data it is told the numbered address of the memory location in which the data lies and it addresses that location and gets the data. Explain further that when the microprocessor saves data it sends the information to a numbered address in memory.
  - Explain that leading from the microprocessor is a series of wires through which the chip addresses (locates) a specific byte (memory location). This set of wires is known as the address bus.
  - Relate that a decision was made to make the address bus of 8-bit microprocessors 16 wires. This means that 2 to the sixteenth power, or 65536 (64K), different locations can be addressed by an 8-bit microprocessor. This is the usual limit of addressable memory in an 8-bit computer. A 16-bit processor has an address bus of 20 lines enabling it to directly address 2 to the twentieth power, or 1,048,580, different locations.
3. Review the function of the ROM chip reminding the students that it contains some of the machine's operating system and often the BASIC language.

4. Indicate that the RAM chips contain a computer's working memory. All programs entered into a computer reside here. All data generated by a program resides here.
5. Relate that a computer's addressable memory is organized and utilized in a specific way.
- . State that as an illustrative example we shall describe the memory organization of a hypothetical computer and that it may or may not be identical to that of a computer the students are using but that in any case the concepts will be the same.

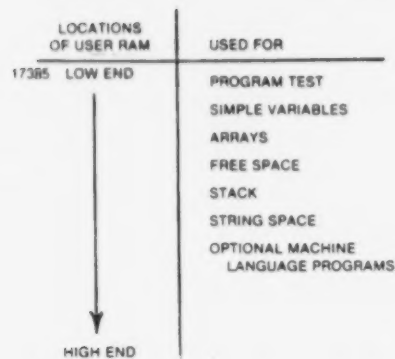
HYPOTHETICAL MEMORY MAP

| DECIMAL ADDRESS     | CONTENTS                          |
|---------------------|-----------------------------------|
| 0                   | 12K ROM (BASIC INTERPRETER)       |
| 12288               | 2K ROM FOR SYSTEM USE             |
| 14336               | KEYBOARD                          |
| 15360<br>→<br>16383 | VIDEORAM                          |
| 16384               | RESERVED FOR SYSTEM USE           |
| 17385               | USER MEMORY FOR PROGRAMS AND DATA |
| 32767               | 16K RAM ENDS                      |
| 49151               | 32K RAM ENDS                      |
| 65535               | 48K RAM ENDS                      |

- . Looking at the chart for our sample machine we see that the memory of this computer is organized in the following way:
  - The first 12K of addressable memory is used for the BASIC language, which resides on a ROM chip.
  - The next 2K of memory is reserved for system use.
  - Following this we find the next higher 1K of memory devoted to interpreting the keyboard signals.
- 6. Explain the following:
  - . Since a computer runs on electricity and we speak English we must translate our English (or BASIC) into electrical impulses which the computer understands.
  - . This is done by translating BASIC into a number code. Cite the following decimal code as an example:

|       |       |
|-------|-------|
| word  | coded |
| ----  | ----- |
| HELLO | 12334 |

- . In the decimal system there are 10 different symbols.
  - . Since in electricity there are two states (on and off) the code we shall use will be based on the binary system, which has two different symbols (0 and 1). We can therefore code our program and data into a series of ones and zeros, which we can then change into a corresponding series of on and off transistors. The computer will electrically manipulate these transistors while doing our task for us. It will then translate the results back into English for us.
  - . A series of eight binary digits is assigned to represent one character (letter, number, symbol, space, or control character). The group of eight bits necessary to represent one character is called a byte. Internally, these bytes are a series of eight transistors, each one representing a bit.
  - . With eight bits a total of 256 ( $2$  to the eighth power) different characters can be represented.
  - . All the manufacturers assign the same number code to the characters that a computer is capable of representing. This representation is called the ASCII Code.
  - . Each keyboard key is located at the intersection of two wires. Pressing a key triggers a unique impulse, which is converted into the ASCII value for that key. This is dealt with by the keyboard RAM section.
7. Following the keyboard section comes the video RAM section.
- . Explain that in order for a computer to print something on the screen it must turn on a pattern of dots on the screen. Each dot on the screen is controlled by a RAM memory location (byte). These dots are called pixels (picture cells).
  - . Indicate that the greater the number of pixels the screen is divided into the greater the screen resolution (that is, the sharper the characters appear on the screen) and the greater the amount of memory needed to control these pixels.
  - . The next 1K of RAM is devoted to system use. The user-available memory begins right after this.
8. Use the diagram at the top of the next page to illustrate the user RAM section organization.
- . Explain that when a program is entered into our sample computer it is stored at the low address section of the user RAM.
  - . The variables generated by the program are stored separately at the next higher set of addresses in our sample machine. Then come the array variables, the stack area (an area utilized to run the program) and finally the area in which string variables are stored.



. Explain that as programs are entered they are stored in their section, thus using up memory from the lower addresses on up, while string variables are placed in their section, thus using memory from the higher addresses on down. The effect, then, is that a computer's user-available RAM resembles an accordion, expanding and contracting from both ends.

9. Refer to the motivation and ask why it is possible to save a program on disk or tape but not the variables.

Elicit that if we write a mailing label program and save it that we will probably want to use it again for making different labels. If we had saved it with the old values we would have to erase them before we used the program for new ones. The manufacturer made things easy for us and set up the save instruction so that it only saves that which is found in the program section of RAM. The variable values, stored in another location, can be saved by writing a program that will specifically save them. This will be presented in a future lesson.

10. Introduce the terms PEEK and POKE. Remind the students of the video screen memory section indicating that each pixel on the screen is controlled by one of these memory locations. Depending on the contents of its memory control section (a byte of memory) a pixel will be illuminated or off. The on or off value is placed there by the program. We can do two things to this byte:

. We can look at the numerical value stored there:

```
PRINT PEEK (1027)
```

which will print the number stored in memory location 1027.

This is of use to us in video games when we wish to know if PAC MAN has been eaten by one of the spooks.

. We can place a numerical value into a memory location:

```
POKE 1027,1
```

which will place a value of 1 into memory location 1027. The 1 will affect the area controlled by byte 1027. Since every machine uses



memory location 1027 for a different purpose there is no way for us to collectively discuss this location.

- . Using POKES can effect very rapid changes on the screen. (One of the favorite uses for it in elementary classes is to change the cursor shape).
- . POKES are also utilized to control functions that are not accessible via the keyboard such as switching on a computer's alternate character sets, setting up a computer's clock display or for rapidly whiting out the screen.

#### Application:

1. Explain how the pattern of a computer's RAM organization makes it possible to save a program but not the data that was generated by that program.

#### Summary:

1. What are the two different types of memory found in a microcomputer?
2. What function does the keyboard RAM section of memory serve?
3. What is a pixel?
4. Why does having a high resolution screen require the use of a lot of memory?
5. Explain what PEEK and POKE statements are.

#### Assignments:

1. Explain how the BASIC programs we enter into the computer are executed by an inanimate, unintelligent machine.
2. What similarities and differences exist between the RAM, ROM, and microprocessor chips?
3. Define address bus.
4. Contrast the difference between high and low resolution graphics.
5. Write a program to white out the screen in two different ways, one using regular graphics and one using POKE statements. Which one is faster? Why?



Assignment for More Advanced Students:

1. Provide the students with the video RAM addresses for their computer. Have them write an efficient program which will help them fill in the following table. (Help them with the POKE values if necessary.)

| RAM address# | PEEK I value | POKE value used to<br>change the screen | Effect |
|--------------|--------------|-----------------------------------------|--------|
|              |              |                                         |        |

## Topic 40

# COMPUTER MEMORY ORGANIZATION AND I/O BUFFERS

Performance Objectives: Students will be able to:

- . draw a block diagram of the main parts of a microcomputer
- . explain what each section of a microcomputer does
- . explain what the operating system does
- . list the normal I/O devices for the microcomputer
- . explain what an interface is
- . explain what a buffer is

Vocabulary: New: operating system, input/output devices (I/O),  
interface (parallel, serial), RS 232C interface  
Review: RAM, ROM, bit, byte, default

Motivation:

- . Place the following on the board:

program:  
-----

run:  
---

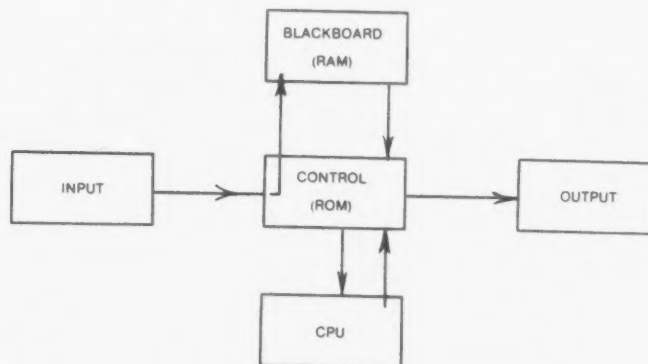
```
10 PRINT "BOB "; BOB BOB B
20 GOTO 10 OB BOB BO
 B BOB BOB
```

Ask what parts of the microcomputer are involved in going from the program to the output.

Aim: How does a computer communicate with its I/O devices?

Development:

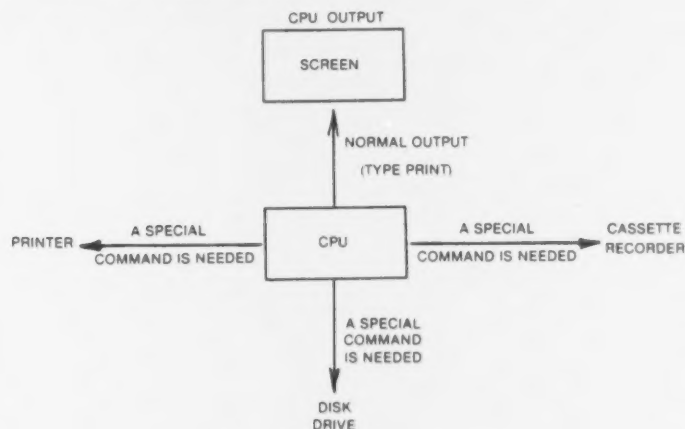
1. Review the block diagram of a computer showing that it has five main sections:



2. Remind students that data is entered into the computer via the input section, directed by the control section (ROM) to the temporary storage section (RAM), then to the CPU for processing. Finally, the results are presented to the user via the output section.
3. Explain that whatever goes on inside a computer is carefully orchestrated by the operating system, which is a set of instructions found in the ROM (control) section that was permanently placed there by the manufacturer. (In some cases the operating system exists on disk.)
4. State that in this lesson we shall describe, in general terms, how a computer communicates with us via its input/output (I/O) devices.
  - . Say that the usual method of inputting information is via the keyboard and that the usual output is to the screen.
  - . Have the students list other input and output devices. Among them should be:
    - printers (for hard copy)
    - microphones
    - card readers
    - magnetic media
    - cassette recorders
    - floppy disk drives
  - . Indicate that while there are other I/O devices we will not specifically deal with them. Stress, however, that many if not all of the concepts to be presented will apply to all I/O devices.
  - . Display both a cassette tape and a floppy disk and say that either can be used for both input and output of data and programs. As a consequence they are often classified as memory devices.
  - . Show a printout and explain that the printer is used for output of data only.
5. Discuss the existence of an interface.
  - . Use the analogy of the three-prong electrical adapter, which allows the insertion of a three-prong plug into a two-slot socket. State that between a computer and its I/O devices there is an adapter called an interface which allows the connection of these devices to the computer.
  - . Indicate that there are two standard types of interfaces, parallel and serial.
  - . Review the concept of a byte, reminding the class that it is made up of eight bits and that in a computer having an 8-bit microprocessor it represents one character.
  - . State that when a computer communicates with an I/O device these eight bits can be sent out at the same time along eight parallel wires (parallel interface) or they can be sent out one at a time along one

wire (serial interface). Mention the RS-232C interface citing it as a serial interface and as an example of an attempt to standardize operations in all microcomputers.

6. State that a computer's CPU must be told by the user where to look for input and where to send the output. Explain that since a user's usual input is the keyboard and the usual output is the CRT screen, the user does not have to specify that these are the desired I/O devices; rather the computer defaults to these.
  - . Elicit that the CPU does not send the information to the screen without instructions to do so; rather, the manufacturer has made it easier for us by making execution of these instructions automatic unless otherwise specified by the user.
7. Explain generally how to specify other I/O devices.
  - . Look at the following diagram. Reproduce it on the board step by step, explaining as you draw. Start with the CPU and its output to the screen and proceed clockwise.



- . Stress that the CPU is connected to the screen, the cassette recorder, the disk drive, and the printer by wires. Compare these wires to tunnels. Show that the output leaves the computer via an open path and that normally only the path to the screen is open. If we type PRINT 9 the operating system routes its results via the only open path, to the screen.
- . Elicit that if the programmer wishes to output to any of the other I/O devices, the path leading to that device must be OPENed. This is done via a line in the program. Elicit further that the opening of these paths is done both for input and for output.
- . Stress the fact that, after use, the paths have to be shut so that the next set of communications can go to the proper I/O device.
- . Mention that even though someone can save a program called "NAME" on a disk or load the program from the disk just by typing SAVE "NAME" or

LOAD NAME" the computer still has to open the "tunnel" to the disk, send the program through the "tunnel," empty the buffer (discussed in item 8, below), and close the "tunnel." Elicit that the manufacturer's instructions in ROM have the computer do all of the above automatically when it detects the proper I/O term.

- . State that the method of saving programs to the disk and loading them from the disk is fairly similar from computer to computer but that there is a big difference in the syntax used for saving and loading data files, a matter that will be dealt with in a future lesson.
- 8. Introduce buffers. Use the analogy of the crowded department store escalator: An area is provided for the crowd spilling off the escalator to move away and collect themselves before stepping onto the escalator to the next floor. Describe how there would be a great congestion without this area.
  - . Remind the class that a computer communicates among its chips at extremely high speeds since all that is moving is electricity. Remind them, also, that the I/O devices have mechanical components and as a result function at much slower speeds.
  - . Indicate that as in the escalator situation an "assembly area" is provided for data going from the computer to the output devices and for the data coming from the input devices to the computer. This area is called the buffer.
  - . Again, use the department store analogy, this time referring to its elevators. Remind the students that the elevator operator usually waits until the car is full before moving out. Explain that so too does data accumulate in the buffer until the buffer is full -- at which point its contents are emptied, either into the output device or, when dealing with an input device, back into the computer.
  - . Remind the students that at closing time, even if the elevator is not full, the operator returns it to the ground floor. Explain that when the end of a program is reached, even if the buffer is not full, it still must be emptied or the data will be lost. Relate that we will learn the special commands that do this in future lessons.
  - . State that in saving a program to disk, the operating system must open a channel to the disk, send the program to the buffer, and then to the disk. Finally, it must close the channel.
  - . Elicit that in loading a program from disk the opposite process is needed.
  - . Mention that there are printer buffers, disk buffers, cassette buffers, and keyboard buffers.

#### Application:

1. Draw a block diagram of a microcomputer, labeling all the sections and using arrows to show the flow of data.

Summary:

1. What directs the operations of a computer?
2. What are the standard input and output devices of the computer?

Assignments:

1. Obtain a picture of the inside of a microcomputer. Identify, as much as possible, the microprocessor, RAM, ROM, and interfaces.
2. Explain what an operating system does.
3. Why do we have to tell a computer specifically to output to the disk drive when we don't have to do the same to output to the screen?
4. Explain the purpose of the buffers found inside a microcomputer.
5. List all the steps that a computer executes in order to get a program transferred from RAM to a disk.

Assignment for More Advanced Students:

1. Write a report on the origin of the RS-232C interface. Include the need for standardization.



**Topic 41**  
**INTRODUCTION TO CASSETTE AND DISK I/O**

Performance Objectives: Students will be able to:

- . define operating system
- . describe disk organization
- . list the steps used in saving information to and recalling information from a disk

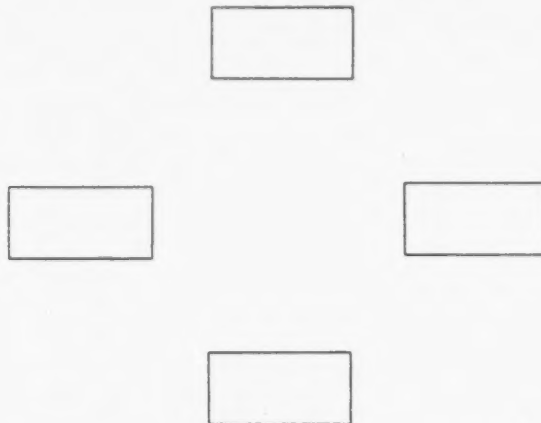
Vocabulary: New: track, sector, DOS, format, disk directory, OPEN, sequential file, random file

Review: buffer, operating system

Aim: How does a computer store information on a disk?

Development:

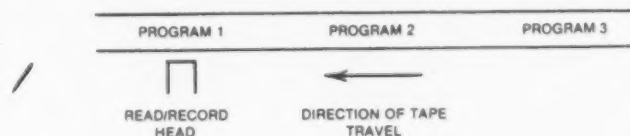
1. Review the block structure of the microcomputer.



- . Locate the following sections: input, control (ROM), CPU (microprocessor), "blackboard - temporary storage" (RAM), output section.
- . Trace the flow of data from its usual input, the keyboard, to its usual output, the CRT.
- . Indicate that all programs and data found in RAM are erased when the computer is shut off. Peripheral storage devices can be used for permanent storage of data. (The students should identify cassette recorders and floppy disks as two such devices).
- . Elicit the fact that a computer's communication with the other output devices is also under the control of an operating system.
- . Remind the students that the CPU is connected to the I/O devices by "tunnels," or channels; that these tunnels have "gates" which must be opened in order for data to pass through to the device and then closed when the CPU is finished communicating with it.

2. Reiterate that a computer's default input is the keyboard and its default output the screen.
  - . Indicate that even though the user does not issue instructions to open and close the channels to these devices the instructions are given behind the scenes by the operating system.
  - . Should the user desire to output to either a cassette recorder or a disk drive this must be specified to the computer.
  - . Explain that the instructions for outputting to the cassette recorder are simple; they open the channel to the recorder, start it, send the information, close the channel, and then stop the recorder.
  - . State these instructions, like all others in the operating system, reside in ROM.
3. Point out that the mechanics of running a disk drive and of handling its associated I/O are very numerous and complex; that the instructions are therefore often located on a disk and loaded into a computer's RAM as they are needed; and that these instructions are called a Disk Operating System (DOS).
  - . Relate that in a disk-based machine there is a small section of code resident in ROM which a computer looks at each time it is turned on or booted. In many machines if a disk is present, this code instructs the computer to obtain from the disk the operating system that is stored there, load it into RAM, and have it take control of the computer.
  - . Explain that DOSs are usually numbered, with the first version often called DOS 1.0. Since the DOS is incredibly complex, and despite all its prerelease testing, some bugs are never uncovered and are inadvertently left in. As users happen upon them or make suggestions for improvements, revisions are made. Minor revisions are identified by a decimal change (for example, DOS 1.1, 1.2). A major revision is indicated by a higher whole number (for example, DOS 2.0, 3.0).
4. Review the concept of the buffer, eliciting the following:
  - . It is a temporary storage location for data being transferred between the CPU and its peripherals.
  - . It is merely an area in RAM set aside for this purpose.
  - . It usually has to fill completely with data before the data moves from it to the peripheral or from a peripheral to the CPU.
  - . If the program is to be terminated before the buffer is full, then specific commands have to be given to empty the buffer and close the channel to the peripheral. The user risks loss of data if this is not done.
5. Review the idea that programs and the data generated by these programs are stored in different locations in a computer's RAM and are handled differently by the operating system.

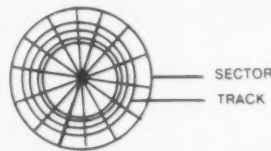
- . Remind the students that listing a program displays the program lines but not the data stored in the variables.
  - . Extend this thought further by eliciting that saving a program on disk or tape would require one type of command while saving the data would require another technique.
  - . List both the cassette and disk program save commands for your computer. Indicate that the techniques for saving data will be covered later in the lesson.
6. Explain that whenever a program or set of data is saved on a disk or cassette it is saved as a unit called a file.
- . Compare the situation to one in which we label a file folder with a program's name, place each program line on a separate piece of paper, put these papers inside the folder, and place the folder inside a file cabinet.
  - . The disk or cassette is analogous to the file cabinet and the program or data file is analogous to the folder that contains the program lines.
7. Explain that information is stored on a cassette sequentially, that data is placed on the tape one byte after the other, and that in order to find something at the end of the tape one must first read everything from the beginning of the tape to that point.



- . Show that because of the nature of the standard cassette's design, this is the only possible way to read and write to cassette.
  - . Relate that when using a disk it is possible to access the information both sequentially and randomly. "Randomly," in this case, means in any order desired.
  - . State that cassette I/O is much slower and less reliable than disk I/O. Remind the students that a tape moves at 1 1/2 inches per second while many disks rotate at 300 revolutions per minute or more.
8. Describe the organization of a floppy disk.
- . Take apart a defective disk and use it for display purposes. Show that it is a circle of plastic with magnetic material secured to its surface. State that the disk drive head is similar to a recorder's head and that the diskette's magnetic particles are oriented by the drive head so that they are magnetically either north or south, representing ones or zeros (bits).
  - . The disk drive receives the program from a computer as electrical impulses and translates the pattern of the impulses (on and off) into

the differing north and south magnetic orientations of the disk particles. When it reads the program from the disk it reads the magnetic particle orientation, translates these orientations into electrical impulses, and sends them to the computer.

- . Stress that, just as electrical impulses serve as a code inside computer chips, so too does the magnetic north-south orientation of the particles serve as a code on the disk.
- . Compare a computer's accessing of information on a disk to the way your letter carrier delivers mail to you. The letter carrier must know not just that you live in a specific house but also what your apartment number is. So too with the disk. It is not enough for a computer to know that a program exists on a disk; it must know the exact location of that program.
- . When a disk is new the user must set up an infrastructure that will allow the addressing of disk material. This process is called formatting a disk.
- . Explain that formatting is done magnetically and that the disk is divided into a series of concentric circles, called tracks, and then further "sliced" into sectors like the wedges of a pie. The number of each sector varies with the brand of computer. Thus files are stored at specific track and sector addresses.



- . Refer to the letter carrier analogy in which each building has a directory listing the tenants and their apartment numbers. The DOS also maintains a two-part directory in which it lists all the files on the disk along with their track and sector addresses. When a person asks for a catalog or directory of the disk, only the names of the files are listed, not their locations.
9. Discuss accessing data files. State that for the programmer the code for saving and loading data files is more complex than that for program files because much more has to be specified.
- . Again, remind the students of the steps necessary to access the disk to save a program file; reiterate that even though they save a program merely by entering the word "save" and the program name, the operating system is adding the other necessary instructions.
  - . List the following as the steps that must be specified by the programmer in writing sequential data files to the disk:
    - reserving buffer space
    - opening the channel to the disk
    - naming the file that the data will be stored in

- telling the disk drive that all subsequent data sent to it should be stored in that file
  - getting the data stored on the disk
  - emptying the buffer
  - closing the file on the disk
  - closing the channel to the disk drive
- . Place the following code on the board, and state that with just this one line we can accomplish the first four items:

```
12 OPEN "O",1,"TEST"
```

or the equivalent OPEN statement for your computer.

- . Indicate that:

'OPEN' opens the channel to the disk drive; this allows us to both send information to and receive information from the disk. We cannot do both in a sequential file, and thus we have to tell a computer which we wish to do.

"O" tells a computer that we wish to output data from the computer to the disk. Stress that all descriptions of the flow of data are relative to the microprocessor and not to any peripheral. Output refers to processor output and input material is flowing into the processor.

'1' Tells a computer that we are setting aside buffer area number 1 to handle the transaction.

"TEST" is the name we are giving to the file.

- . Explain that, so far, all we have succeeded in doing is laying the groundwork for writing to the disk and that the actual mechanics will be learned in the next lesson.

#### Application:

1. Write the code that will use buffer 3 to write data to the disk and will store the data in a program called TEST1. (OPEN "O",3,"TEST1" or your equivalent.)

#### Summary:

1. What is an operating system?
2. How is a disk organized?
3. What is formatting?

#### Assignments:

1. Describe what an operating system is and what a disk operating system (DOS) is. Explain how they are similar and how they differ.

2. Explain sequential storage.
3. Explain how a disk is formatted and how this allows data to be easily stored and recalled from it.
4. List the steps necessary for the programmer to code in order to save information to and recall it from a disk.
5. Compare the use of a cassette tape to use of a floppy disk as a means of permanent storage. Explain how data can be stored on them and list the advantages of using the disk as opposed to the tape.

Assignment for More Advanced Students:

1. Do additional reading and then explain the following aspects of disk storage: single density versus double density, tracks, sectors, granules, files, records, and bytes.



## Topic 42

### SEQUENTIAL DISK FILE STRUCTURE

Performance Objectives: Students will be able to:

- . write a program that will create a data file and store it on a disk
- . write a program that will input the contents of a disk data file and print it on the screen

Vocabulary: New: PRINT #, CLOSE, INPUT #  
Review: sequential file, OPEN

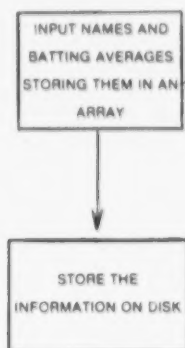
Aim: How can we write a program that will read from and write to a disk file?

Development:

1. Review the concept and use of files.
  - . Have students distinguish between a program file and a data file.
  - . Have students distinguish between a sequential file and a random access file.
  - . Have students list the steps needed in order to save a program file on a disk.
  - . Reinforce the idea that even though a program is saved merely by entering the word "save" and the program's name, other steps are necessary and the operating system adds them.
  - . Have students list the steps necessary in order to save a sequential data file on the disk. They should include:
    - reserving a buffer to handle the transaction
    - opening the channel to the disk
    - naming the file that the data will be stored in
    - telling the disk drive that all subsequent data sent to it should be stored in that file
    - getting the data stored on the disk
    - emptying the buffer
    - closing the file on the disk
    - closing the channel to the disk drive
  - . Have students write the one program line which will accomplish the first four items. (12 OPEN "O",1,"TEST" or your equivalent OPEN statement)
  - . Have students explain what each part of the statement does. They should include the following:
    - "OPEN" opens the channel to the disk drive allowing us to both send and receive information from the disk.
    - We cannot do both in a sequential file, and we thus have to tell the computer which operation we wish to perform.

- "0" tells the computer that we wish to output data from the computer to the disk.
  - "1" tells the computer that we are setting aside buffer area number 1 to handle the transaction. (There are many such buffers that we can use.)
  - "TEST" is the name we are giving to the file.
- . Explain that so far all that has been done is to lay the groundwork for writing to the disk. A mechanism by which the writing can be done is still needed.
2. Indicate to the students that to learn about sequential data files they will write a program that inputs the names of three baseball players and their batting averages, storing them first in an array and then in a sequential file on the disk.
- . The students should be encouraged to write in modules. Thus the inputting and storing in an array could be done in one module and the disk storage could be accomplished in another module.

Elicit the following flow chart:



- . Have the students write the first module:
- ```

5 CLS (HOME or SHIFT (CLS/HOME) )
10 REM: Reserve string space command (CLEAR 100 in TRS-80)
20 FOR I= 1 TO 3
30 INPUT "ENTER THE PLAYER'S LAST NAME, FIRST NAME AND BATTING
   AVERAGE - USE NO COMMAS"; A$(I)
40 NEXT I
  
```
- . Remind the students that if the names and data they enter are too long they will run out of string space and will have to reserve additional spaces with the CLEAR statement (if their computer requires preallocation of string space).
3. State that they are now ready to save to the disk and elicit that they should begin with the OPEN statement.
- . Again, review the things that the OPEN statement accomplishes:

- opens the channel to the disk drive
 - tells the computer that we wish to output data to a sequential file ("O")
 - indicates which buffer will be used (1)
 - names the file that will store the data
- . Have the students write the OPEN statement for the program, using buffer number 2 and naming the file BASEBALL. (50 OPEN, "O",2, "BASEBALL" or equivalent)
 - . Elicit that the above line laid the groundwork but did no writing of data to the disk.
 - . Have the students write a module which would print the input data on the screen.

Acceptable answer:

```
60 FOR I=1 TO 3
70 PRINT A$(I)
80 NEXT I
```

- . Show the students that to have the names stored on the disk all they would have to do is change line 70 to read:


```
70 PRINT #2, A$(I) (or your equivalent PRINT to disk statement)
```
 - . Explain that:
 - PRINT # directs the output to the disk.
 - "2" tells the computer that buffer #2 will be used.
 - . Caution the students that this number must match the number in the OPEN statement and that there must be a comma after the PRINT #2.
 - . State that the PRINT #2,A\$(I) statement places the contents of A\$(I) on the disk followed by the code for a carriage return, which acts as a separator (delimiter) for items on the disk.
4. Have the students review the program as written so far. Have them identify those lines that are unique to disk I/O. (line 50, the OPEN statement and line 70, the PRINT # statement)
 5. Elicit that the program is complete save for the closing of the buffer and of the channel to the disk drive.
 - . Add the following:


```
100 CLOSE #2 (or your equivalent CLOSE statement)
```
 - . Explain that this statement does four important things:
 - It sends the buffer's contents to disk. (Remind the students that on many machines data is transmitted to the disk from the buffer only when the buffer is full; should the program end with the buffer being

partially full then the data would be lost unless the buffer was flushed. The CLOSE statement does this.)

- It releases the buffer for reuse. (A sequential buffer can be used for only one job, either input or output to one file but not both at one time. The CLOSE statement releases it for other duty.)
- It closes the file on the disk. (If a file is not closed and the disk is then reaccessed, losses of data in the entire disk may occur.)
- It closes the computer's output channel to the disk drive.

- . Indicate that CLOSE #2 just closes buffer #2. Point out that it is a good idea to just write CLOSE which will automatically close all open files, buffers and I/O channels.
- . Show that, except for the beginning, much of what goes on in the above program occurs in the circuitry and that very little is seen on the screen, and so students may have the feeling that nothing is happening. It is a good idea, therefore, to add PRINT messages such as "NOW SAVING" before the saving commences.

- . Stress that when a statement like

OPEN "O",1,"BASEBALL"

is executed the DOS checks the diskette's directory to ascertain whether or not a file called BASEBALL already exists.

- If no file exists with that name, a new one is created and its existence is noted in the directory.
- If such a file already existed, the DOS would cause the old file's directory entry to be written over with the new information about the file BASEBALL, effectively wiping out the old file that had that name.
- . State that OPEN "O" cannot be used to add data to an existing file. Special append versions of OPEN exist for many versions of BASIC and they are all different.

6. Ask the students about a program that will retrieve the above data from the disk and print it on the screen.

- . Elicit that the first step would be to write an OPEN statement.
- . Point out that the previous OPEN statement read OPEN "O", which told the computer that the students wished to OUTPUT data from the computer to the disk. Now, however, they have to tell the computer that they wish to INPUT data from the disk to the computer. The students should provide the following OPEN statement:

```
110 OPEN "I",1,"BASEBALL" (or your equivalent)
```

- . Make the following points:

- The OPEN statement opens the channel between the computer and the disk drive.
- The "I" tells the operating system that we are to retrieve data from the disk.

- The 1 tells the operating system that we are assigning buffer number 1 to handle the transaction.
- "BASEBALL" is the name of the file we wish to access.

- . Have the students recall the module they wrote that took the data from the array and printed it on the screen.

```
60 FOR I=1 TO 3
70 PRINT A$(I)
80 NEXT I
```

- . Remind them of the change they made that sent the data to the disk.
(PRINT# ")
- . Explain that here, too, only a minor change has to be made to get the data from the disk into the computer's RAM, that is:

```
INPUT #1, A$(I)
```

- . List the following module on the board.

```
110 OPEN "I",1,"BASEBALL"
120 FOR I = 1 TO 3
130 INPUT #1, A$(I) (or your equivalent)
140 NEXT I
```

Trace the program eliciting that while the INPUT #1 statement gets the data from the disk (through buffer #1) it only stores it in an array. In order to have the data printed on the screen we must add another line that does this.

- . Elicit line:

```
135 PRINT A$(I)
```

- . Finally, elicit that a CLOSE statement is needed:

```
150 CLOSE
```

7. Test the student's understanding of the program by asking the following:

- . The data was originally stored on the disk using buffer #2. How can the data be retrieved from the disk using buffer #1?
- . Is it necessary for the CLOSE statement to come after the NEXT statement? If not, why is it placed there?
- . Why is it better to write CLOSE and not just CLOSE 1?

Application:

1. Write a program that inputs the names of the five boroughs making up New York City from a data statement and uses buffer #1 to store the data in a sequential data disk file called BOROUGHs.

Acceptable answer:

```
10 FOR I = 1 TO 5
20 READ A$(I)
30 NEXT I
40 OPEN "O",1,"BOROUGHs" (or your equivalent)
50 FOR I = 1 TO 5
60 PRINT #1, A$(I) (or your equivalent)
70 NEXT I
80 CLOSE (or your equivalent)
90 DATA "BROOKLYN", "BRONX", "MANHATTAN","QUEENS","STATEN ISLAND"
```

Summary:

1. What does the OPEN statement accomplish?
2. What does the PRINT # statement accomplish?
3. What does the INPUT # statement accomplish?
4. What does the CLOSE statement accomplish?

Assignments:

1. This problem extends the application problem. Write a program that will input the names of the five boroughs of New York City from data statements, store them in a sequential data file on disk, close the file, and then input the names from the disk printing them on the screen.
2. Write a program that will create a sequential data file to save to disk a student's grades in the five major subjects. Each entry will consist of the subject name separated from the grade by a dash. The information is to be entered into an array via READ/DATA statements.
3. Expand the above program so that after it stores the data on the disk it also stores the program itself on the disk.
4. Add program code to the above to input the data saved and print it on the screen.
5. Expand further on the above two programs so that besides inputting and printing the stored data the program is also printed.
6. Find out the command to save a data file to a cassette tape. Rewrite the answer to the first question (the borough question) to both store the data on a cassette and then retrieve it, finally printing it on the screen.
7. Write a program that INPUTs five names and saves them on disk in a sequential file called TEAM.
8. Write a subroutine that will obtain the names from disk and PRINT them on the screen.

Assignments for More Advanced Students:

1. Write a program that will input the last name of a group of people into one array, their first names into another array, and their telephone numbers (including the area code but no dashes or spaces) into a third array. The names can be any length upon input but the program will limit their length to 10 letters for the last name and 6 letters for the first name. It will limit the telephone number to 10 characters when it stores them. The names should be truncated or have blanks added to them as necessary. They should then be stored in a fourth array from which they will be transferred to a sequential file on a disk.
2. Write a program that will retrieve the data stored by the above program and print it on the screen.

Topic 43

CAI/CMI PROGRAMS

Performance Objectives: Students will be able to:

- . write a CAI program implementing many of their previous learnings
- . write a CMI program

Vocabulary: New: CAI, CMI

Motivation:

- . Administer the following \$64,000 quiz program:
 - How much is 20 times 8?
 - What is the capital of Madagascar? (Tananarive)
 - Does an adjective modify a noun or a verb? (a noun)
 - What elements are found in rust? (iron, oxygen)
 - How much is $1/4$ of $1/2$? ($1/8$)
 - What state borders western New York City? (New Jersey)
 - What is the lightest chemical element? (hydrogen)
 - What is the French word for door? (porte)
 - Which is correct... "I did good" or "I did well"? (I did well)
 - What is the French for "I go"? (je vais)
- . Ask the students the following:
 - How many questions were answered correctly?
 - In which category were all the questions answered correctly?
 - In which category were all the questions answered incorrectly?

Elicit that it would have been better had it been announced at the beginning that a tally should be kept.

Elicit that using a computer to administer the test and keep a running tally would have been best of all.

Aim: How can we use a computer to help students learn?

Development:

1. Ask the students to describe what they consider the steps involved in the teaching of new material. Elicit that proper teaching involves many things including presentation of new material, drill and practice problems, and testing.
 - . State that a computer can be used to do all of the above but that we will deal mainly with drill and practice problems and testing.
 - . List the following four kinds of drill and practice problems that a computer can be used to present:
 - problems alone

- problems with answers
 - problems that provide the correct answers and the reasoning
 - problems which lead the student to the solution through a method of Socratic questioning
- . Elicit that a computer can be used to provide all these types of problems. The last two are the most desirable.
 - . Refer to the motivation and elicit that the value of the drill and practice problems would be enhanced if the computer recorded the number and kind of problems the student got wrong and then stored the information on a disk. The teacher could monitor the student's progress in great detail, which could not easily be done in the conventional classroom.
 - . Elicit further that a computer could also be instructed to provide similar problems until they were answered correctly. If this type of approach were to be combined with the presentation of the solution through the Socratic method, the student would be the beneficiary of a personal tutor.
 - . Elicit that a computer could also be programmed to ask for a student's name and could then address statements to the student by name. State that this seems to humanize the computer and to spark an interest among youngsters.
 - . Elicit that a computer could also print an instant score and diagnostic record for the student to take along and at the same time print a master overall record for the teacher. Of course the set of records would be kept on a password-protected disk so that each student could access his or her own but no others.
2. The computer could also be used to present difficult material in novel ways. The machine could accomplish this using its graphic capability and its ability to execute many sequential steps adding one new bit of information at each step to make the whole screen seem new.
 - . Cite this assignment for a class learning about the geography of New York City: An unlabeled map of the city was distributed and the students asked to label all the boroughs, the bodies of water, and the lands bordering the city. Elicit that a computer might be used in the following ways:
 - Printing a map on the screen labeling the areas to be identified. (Indicate that using a computer in this manner would be very wasteful. A sheet of paper would be much more effective.)
 - Printing a series of maps showing different aspects of the city, including, for example, a subway map, a map of the bridges and tunnels, a map of the airports, a map of the parks, a map of business and population centers, a map of the beaches, and a map of the museums in the city.
 - . Elicit that a computer could thus take the place of a series of test maps and also stimulate interest by:

- sequentially displaying individual items, erasing the old label and displaying a new one
 - highlighting an item by flashing it on and off
 - printing an item to be identified in a color other than the predominant color
3. Have the class spend five minutes writing a drill and practice program that would provide an unlimited number of simple addition problems for a beginning math student. It would ask for the student's name and do nothing other than generate the problems, indicate whether the answer was correct or incorrect, give the correct answer if needed, and allow the student either to get more problems or to terminate the program.

Acceptable answer:

```

5 CLS : REM OR HOME OR SHIFT CLR/HOME
10 PRINT "PLEASE TYPE YOUR NAME AND THEN PRESS <CARRIAGE RETURN>"
20 INPUT A$
30 PRINT "HELLO..."; A$
40 PRINT "I WILL GIVE YOU SIMPLE ADDITION PROBLEMS TO DO"
50 LET A = RND(10) (use your computer's random generator)
60 LET B = RND(10)
70 PRINT "WELL ";A$; " WHAT IS THE SUM OF ";A;"+";B;"?"
80 INPUT C
90 IF C = A + B THEN PRINT "CORRECT"
100 IF C <> A + B THEN PRINT "WRONG ... THE CORRECT ANSWER IS " ; A+B
110 PRINT "PRESS <C> TO CONTINUE THE QUIZ OR <CARRIAGE RETURN> TO END"
120 INPUT B$
130 IF B$="C" THEN GOTO 5
140 PRINT "THANKS ";A$;" THAT WAS FUN"
150 END

```

4. Ask the students what would have to be done to get the computer to do the following:
- . provide a choice of addition or multiplication problems (add a statement asking for the choice, an INPUT statement, a GOTO, a multiplication module).
 - . keep score of the correct and incorrect responses (add counter statements -- one after the correct response IF test (line 90) and another after the incorrect response IF test (line 100)).

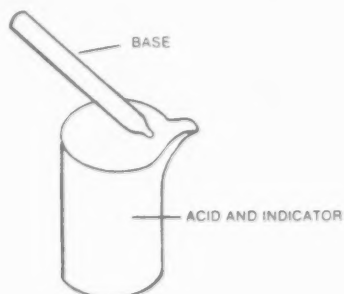
Point out that the program could have been changed so that it allowed more than one chance at the correct answer. (Do not show how; this assignment will be part of the homework.)

5. Indicate that the above examples were simple ones and that more complex problems can be generated for, among others, algebra, physics, and chemistry classes.
- . Discuss the value in this case of providing more than just the correct

answer: A student getting the wrong answer might benefit more from having the solution revealed in steps rather than just having the answer given. Cite as an example the determination of the area of a square 2 cm. long. Should the student get the answer wrong the computer could first give the formula for determining the area of a square and then give the student a chance to answer. If the student still gets the answer wrong the computer could then teach the student, in steps, how to solve the equation.

6. Discuss computer-generated simulation problems. Indicate that this is a very important way in which a computer can be used to do things that would be impossible otherwise.
 - . Cite the training of astronauts, airplane pilots, and nuclear reactor operators.
 - . Discuss a chemistry program simulating an experiment in which the quantity of acid in a beaker is measured by titration (adding an indicator that changes color and adding a material that will combine with the acid). When all the acid is combined with the base the indicator changes color and the amount of base that was added is noted. A formula is used to calculate the amount of acid present.

In the program, graphics should be used to draw the acid and base containers. The program chooses the amount of acid to be neutralized and then changes the color in the container when the proper amount of base is added. (For black and white computers different printed symbols could be used to accomplish the same result.) All of this, of course, is done on the screen. The student is then asked to calculate how much acid was present. If the wrong answer is given the student is sequentially provided with pieces of the solution and allowed to try to answer again until, finally, the entire solution is presented along with the correct answer.



7. Name the two main ways in which the students have seen computers used as drill and practice tools.
 - . CAI - computer assisted instruction (A computer generates problems and checks the answers.)
 - . CMI - computer managed instruction (A computer not only generates the problems but also keeps track of the number of correct and incorrect responses. It can also keep a record of the kinds of problems the student got wrong.)

This information could be used in two ways:

- It could generate more of this type until the student gets them correct.
- A computer could file these records on a disk and thus be capable of providing, on demand, a complete progress report and/or diagnostic record for each student.

These files could be password protected so that only the proper people would have access to them.

Application:

1. Expand upon the program used in class by having it count the number of incorrect responses, storing the information in a sequential disk file when the program is over. The file's name should be the student's name.

Acceptable answer:

Add the following lines:

```
to the end of line 100: : I=I+1
141 PRINT "NOW STORING..."
143 OPEN "O",1,A$           (or your equivalent)
145 PRINT #1, I             (or your equivalent)
147 CLOSE                  (or your equivalent)
```

Summary:

1. What are two ways in which a computer can be used to provide drill and practice in math.
2. Explain what CAI is.
3. Explain what CMI is.
4. Explain how using disk files to store records improves the quality of CAI.

Assignments:

1. Write a modular CMI drill and practice program that will:
 - . provide an unlimited amount of math problems
 - . ask for the student's name, and address the student by name in subsequent requests
 - . provide a set of easy addition problems (dealing with the numbers 1 to 10) for the student
 - . check the answers, printing the words "correct" or "incorrect" (giving the correct answer) as the case might be

2. Add to the program so that:

- . The student has a choice of easy (dealing with the numbers 1 to 10) or hard (dealing with 50 to 100) problems

- . The student will be given two chances to give the correct answer

3. Add to the program so that it keeps track of the student's score and of the kinds of problems answered right or wrong (addition, subtraction, easy, hard).

Assignments for More Advanced Students:

1. Add to the program so that it stores all records on a disk and prints out a diagnostic record on demand.

2. Add a timer to the above program so that a student has a given amount of time to answer before he or she is considered to have given a wrong answer.

3. Write a program that presents an algebra, physics, or chemistry problem. If the wrong answer is given the computer will tutor the student step by step, first giving the formula to be used and then adding parts until, finally, the correct answer is given.

Topic 44

USING STR\$ AND VAL IN A PROGRAM

Performance Objectives: Students will be able to:

- . define subroutine
- . define function
- . define library function
- . use the STR\$ function to convert numbers to string variables
- . use the VAL function to convert strings to numbers

Vocabulary: New: subroutine, function, STR\$, VAL, DEF FN, library function
Review: variables, INPUT, GOSUB

Motivation:

- . The following program is run:

```
10 INPUT A$
```

As input a person enters a 1. Will the computer accept it as input?
(Yes; it accepts it as string input, not as a number, and prints an error message.)

- . The following program is run:

```
10 INPUT A
```

As input a person enters an A. Will the computer accept it as input?
(No; it will reject a string.)

Aim: How can we use VAL to avoid a redo error in a menu-driven program?

Development:

1. Review the concept of a variable.
 - . Have the students think of a variable as a container in the computer's memory that can hold any value.
 - . Indicate that there are two types of variables: string variables, which can hold letters, symbols, and numbers (but the numbers exist only as patterns on the screen and therefore cannot be used for doing math); and numeric variables, which can hold only numbers (and the computer can do math with them).
2. Review the INPUT statement.
 - . Have the students indicate that it halts program execution until something is entered into the computer via the keyboard. Make sure they understand that:
 - INPUT A\$ takes the keyboard input and assigns it as a value to A\$

- Anything on the keyboard can be assigned to A\$ (even a carriage return -- the null value)
 - Numbers are assigned as string values (math can't be done with them)
 - . Explain that when a computer executes the INPUT A statement, it expects a number as input, and if it gets anything else (even a numeric variable) it responds with an error message and repeats the ? prompt of the INPUT statement.
3. Review the concept of a subroutine.
- . Indicate that it is a programming module which is independent of the main body of the program.
 - . A subroutine can be called (have data sent to it for processing) from any point in the program.
 - . When the subroutine finishes processing the data, the machine returns to the statement following the GOSUB which called it.
 - . Contrast this with the GOTO statement.
 - . Functions are a kind of subroutine. Functions and other subroutines differ in the way they are called.

Subroutine:

Place the following example on the board:

```
10 PRINT "GO"
20 GOSUB 70
30 PRINT "TEAM"
40 GOSUB 70
50 PRINT "GO"
60 END
70 REM SUBROUTINE
100 PRINT
110 RETURN
```

Show that the subroutine, which in this case consists of one command, the PRINT statement (but which can consist of many lines), starts at line 70 and ends at line 110.

The subroutine is called by the GOSUB statement in lines 20 and 40.

The results are sent back to the part of the program that called it by the RETURN statement.

Functions:

Show that, in our first example, we used the subroutine to print a

blank line. We used the word GOSUB to send the program there and the word RETURN to send the program back to where it began.

Contrast the above with the following:

```
10 DEF FNA(X) = X*X*X      (or your equivalent)
20 PRINT "WHAT NUMBER DO YOU WISH TO CUBE"
30 INPUT X
40 Y = FNA(X)
50 PRINT "THE CUBE OF "; X ; " IS" Y
```

Demonstrate the following:

- The function was set up in line 10 using the DEF FN statement.
- The function was called in line 40. Show that a GOSUB was not used to accomplish this. Explain that using the DEF FN and the FN A statements causes the computer to look in its memory for the function labeled A and to transfer control to it. At the same time, when the function is finished the computer knows to transfer control back to the sending line. Show that no RETURN statement is found in the program.

Library functions:

Explain that the authors of the BASIC language felt that there would be a frequent need for certain calculations and data manipulations so they included the code for doing them as functions as part of the BASIC interpreter in ROM. A programmer, therefore, does not have to define these manipulations before using them; rather they simply must be called. Such functions are called library functions as contrasted with "user-defined" functions.

Have the students list some library functions they have used. Among these could be:

```
MID$
LEFT$
RIGHT$
LEN
```

Indicate that in this lesson the students will learn about two other library functions, STR\$ and VAL.

4. Discuss menu-driven programs and VAL. Place the following on the blackboard as an example of a menu:

```
10 PRINT "TYPE THE NUMBER WHICH LIES NEXT TO YOUR CHOICE"
20 PRINT "<1> TO ENTER NAMES"
30 PRINT "<2> TO SORT THE NAMES"
40 PRINT "<3> TO CHANGE THE LIST"
50 PRINT "<4> TO EXIT THE PROGRAM"
```

- . Explain that in menu-driven programs the user chooses from a list of

options indicating the selection by entering the number corresponding to that choice.

- . Elicit that the next line in such a program might be an INPUT statement.
- . Ask the students whether it should be INPUT A\$ or INPUT A.
- . Demonstrate the advantage of using INPUT A; i.e., that by entering the choice as a number the ON A GOTO statement could be used. This is a very powerful statement whose use would save a number of program lines as in the following example:

```
100 INPUT A
110 ON A GOTO 1000,2000,3000,4000
```

versus:

```
100 INPUT A
110 IF A = 1 THEN 1000
120 IF A = 2 THEN 2000
130 IF A = 3 THEN 3000
140 IF A = 4 THEN 4000
```

- . Discuss the disadvantages of using INPUT A. Indicate that if, in response to the INPUT A statement, the user types ONE instead of 1, the computer will respond with:

```
? REDO
?
```

This message is unclear for it does not explain to the user what the error was. The user is likely to go on repeating the mistake for a while.

Elicit that the problem would not exist if we used an INPUT A\$ statement. There would be no REDO message and the computer would not be caught in a loop because INPUT A\$ accepts anything as input.

Indicate that we gained by avoiding the REDO error but we lost because we cannot use the ON A GOTO statement. Ideally we should be able to do both.

State that by using the library function VAL we can use both the INPUT A\$ and the ON GOTO statements.

- . Place the following on the board:

```
10 PRINT "ENTER ANY NUMBER FROM 1 TO 4"
20 INPUT A$
30 V=VAL(A$)
40 ON V GOSUB 100,119,230,432
```

Elicit that VAL is a function that converts string values into their numeric equivalent.

Explain the above program showing how the VAL function has converted the string input into numeric values (V).

- . Have the students suggest a method for dealing with a user who types ONE as input.

Hint: What is the value of A when the following program is executed:

```
10 PRINT A
```

The program would print an 0 since it was never assigned a value and its default value is 0. Have them relate this to the question.

Acceptable answer:

```
35 IF V<1 OR V>4 THEN PRINT "ENTER 1,2,3 OR 4 PLEASE" : GOTO 10
```

Elicit that this line works because the VALue of a word is 0.

5. Discuss STR\$. Indicate that while the VAL function converts string variables to their numeric equivalents the opposite is done by the STR\$ function.

- . Place the following on the board:

```
10 A$ = STR$(2)  
20 PRINT A$
```

Ask what would be printed when the program is executed. (2)

Add two more lines:

```
30 IF A$="2" THEN PRINT "CORRECT"  
40 IF A$<>"2" THEN PRINT "WRONG"
```

Relate that when the program is executed the computer will print WRONG. Explain that when STR\$ converts a number into a string it includes the leading blank that normally preceeds a number so that the comparison had to be IF A\$=" 2".

Application:

1. Write a menu module for a CAI math drill program which asks you to choose among addition, subtraction, multiplication, and division problems. After the selection the menu directs the program to line 100 for addition, line 175 for subtraction, line 231 for multiplication, and line 82 for division.

The choices must be those of numbers input as strings and the ON GOSUB statement must be used to direct the program flow.

Acceptable answer:

```
10 PRINT "ENTER THE NUMBER PRECEEDING YOUR DRILL CHOICE"
20 PRINT "<1> ADDITION"
30 PRINT "<2> SUBTRACTION"
40 PRINT "<3> MULTIPLICATION"
50 PRINT "<4> DIVISION"
60 INPUT A$
70 A=VAL(A$)
80 ON A GOSUB 100, 175, 231, 82
```

Summary:

1. What is a variable?
2. What is a subroutine?
3. What are the two types of subroutines we dealt with?
4. What is a library function?
5. What does the VAL function do?

Assignments:

1. Write a program that allows the input of five numbers into string variables, converts these strings into numbers, and then reconverts them back into strings. The program's output will be in five columns and three rows. The first row will contain the original set of numbers, the second row the converted numbers and the third row the reconverted string numbers.
2. Write a program that defines a function to multiply two numbers together, divide the result by 2, find the integer value, and add 5 to that. The main program will ask the user to input two numbers and will call the function. After the result is obtained and printed, the program will ask if the user wishes to enter new numbers. If the answer is YES the computer will allow their input. If the answer is NO the computer will print a message saying THANK YOU FOR LETTING ME SHOW OFF.
3. Write a menu module that will ask the user to choose from among five items. The choice will be made by choosing one of five numbers. The numbers will be INPUT into a string variable and then used in an ON GOTO statement. There should be code to ensure that any input other than the listed numbers causes the menu to be repeated.

Assignment for More Advanced Students:

1. Write a program that uses READ/DATA to enter 26 numbers into a program, converts them into strings, and removes the leading blanks.

Topic 45

USING ASC AND CHR\$ WITH FILE MANAGEMENT PROGRAMS

Performance Objectives: Students will be able to:

- . explain the ASC function
- . explain the CHR\$ function
- . use the CHR\$ function to create loops controlled by letters instead of by numbers

Vocabulary: New: ASC function, CHR\$ function, control characters

Review: library function, ASCII Code, OPEN, PRINT #, buffer, INPUT #

Motivation:

- . A program requires the input of an item of data five characters long. What code can Janet write that would print a line of five dashes and then move the cursor back to the beginning of the line ready for the character input?

□ _ _ _ _ _

Aim: How can we use the CHR\$ and ASC functions to enhance a program?

Development:

1. Review the concept of the ASCII Code.
 - . We communicate with the computer by using a binary number code.
 - . In an attempt to effect a measure of uniformity in the computer field certain number code values have been standardized. Among these are all the keyboard entries. This means, for example, that when the letter A is pressed on any microcomputer the CPU receives a code value of 65.
 - . This code is called the ASCII Code.
2. Review the concept of subroutines and functions.
 - . Have the students list the library functions they are already familiar with, including STR\$ and VAL.
 - . Elicit that they are opposite functions -- that one undoes what the other does. STR\$ converts a number into a string variable while VAL converts a string variable into a number.
 - . Indicate that in today's lesson students will learn two other library functions, ASC and CHR\$, and that these also are opposites.
3. Place the following program on the board:

10 PRINT ASC("A")

- . Define ASC as a library function that deals only with strings and that returns the ASCII Code equivalent of the item inside the parentheses. The item must be in quotes.

When executed, the program will print a 65 on the screen, the ASCII Code number for a capital A.

- . Elicit that while the code for a capital A is 65, the code for lowercase a is different.
- . Stress that the item in the parentheses must have quote marks around it. Illustrate this by pointing out that

PRINT ASC(1) returns a Type Mismatch ERROR

while

PRINT ASC("1") returns a 49

4. Put the following on the board:

```
10 PRINT CHR$(65)
```

- . Define CHR\$ as a library function which does the opposite of what ASC does, that is, it changes an ASCII Code number into its string equivalent.
- . Elicit that the above program, when executed, would PRINT an A on the screen.

Show the following relationship:

ASCII CODE	CHARACTER
65	A
PRINT ASC("A")	PRINT CHR\$(65)

5. Discuss CHR\$ and control characters. Explain that there are code values not only for letters, numbers (as strings) and symbols but also for keys that do not print anything on the screen but rather cause something to happen. Elicit that among them are such items as:

- . shift keys (for capital to lowercase and vice versa)
- . cursor movement keys
- . carriage return keys (ENTER or RETURN)
- . screen clearing keys
- . break keys
- . printer codes
- . letter size changes (on the TRS-80, for example)
- . Illustrate this by referring to the CLS command of the TRS-80. (Use the key(s) appropriate for your computer.) Demonstrate that while the

clear key can be pressed when entering program lines in the immediate mode it is functionless while a program is executing; hence the use of a program command to clear the screen. State that while there is a CLS command that accomplishes the same thing as pressing the clear key, there are no special commands for many of the other action keys. Thus we have the use of the ASCII Code to achieve the desired action.

- . Elicit that to effect use of the code one would type:

```
PRINT CHR$(the code of the desired action goes here)
```

- . Refer to the Motivation. Place the following diagram on the chalkboard:

```
> RUN
ENTER 5 LETTERS
? -----
```

- . Elicit that (1) the ? was placed on the screen by the INPUT statement, and (2) the stationary cursor indicates that the machine is awaiting INPUT.
 - . Tell the students that, when something is printed on the screen, using the backspace key erases material that was already typed. Ask students how it was possible to print five dashes and then get the cursor to return to the beginning of the line without erasing the dashes. (Printing control characters is the key to this operation).
6. Explain that another use for the CHR\$ function would be to include quotes inside a string. Show the following:

```
PRINT "GO WEST YOUNG MAN SAID HORACE GREELEY AND WALTER O'MALLEY LISTENED"
```

- . Elicit that any quote marks put around the GO WEST YOUNG MAN would serve to terminate the string and it would not be printed. The quotes can be printed if the statement is written with the CHR\$ code for a quote as follows:

```
PRINT CHR$(34); "GO WEST YOUNG MAN"; CHR$(34); "SAID HORACE GREELEY AND WALTER O'MALLEY LISTENED"
```

This would PRINT:

```
"GO WEST YOUNG MAN" SAID HORACE GREELEY AND WALTER O'MALLEY LISTENED
```

7. Discuss the use of CHR\$ in graphics. Indicate that, besides using the PRINT CHR\$() for representing alphanumeric and control characters, many computers use it for representing different character types (such as the Japanese characters of some TRS-80 models) and also for representing graphic characters not accessible through the keyboard on many machines.
8. Discuss the use of CHR\$ to open disk files. Review the syntax for opening a sequential disk file named A in order to store items in it on disk.

OPEN "O",1,"A" (or your equivalent)

Have the students identify each part of the statement:

OPEN - opens the channel to the disk
"O" - opens the channel for OUTPUT to the disk.
1 - assigns buffer #1 to handle the data.
"A" - is the name of the file on the disk.

. Pose the following problem to the students:

- Write a program that opens 26 different sequential files and stores the name of the file in each one. Have the program consist of no more than five program lines (not including an END statement) with no more than a single statement per line.

After giving them a few minutes to think, elicit that three of the lines would be directly related to file management:

an OPEN statement	10 OPEN "O",1,-----
a PRINT # statement	20 PRINT #1,-----
a CLOSE statement	30 CLOSE

- . Elicit that a mechanism is needed whereby 26 files are OPENed and whereby the file names can be stored in the file.

Suggest that a FOR/NEXT loop could be used to open the 26 files.

While FOR I=1 TO 26 would open 26 files, it would not meet the requirement that there be only five single-statement program lines; ask how the students would name the file and store the name in the file given those conditions.

If no suggestions are tendered, explain that using the information in today's lesson would enable them to solve the problem.

If still no solutions are offered, remind the class that there are 26 letters in the alphabet and that these should be the names of the files. Challenge them to use either the ASC or CHR\$ function to solve the problem.

- . Elicit the following solution:

```
5 FOR I=65 TO 90
10 OPEN "O",1,CHR$(I) (or your equivalent)
20 PRINT #1,CHR$(I) (or your equivalent)
30 CLOSE
40 NEXT I
```

Application:

1. Write a program that will display the character string equivalents for the ASCII Code numbers from 1 through 255 along with their code numbers.

Acceptable answer:

```
10 CLS
20 FOR I=1 TO 255
30 PRINT I;" ";CHR$(I);
40 NEXT I
```

Summary:

1. What does using the ASC("Z") function accomplish?
2. Explain what the CHR\$() function does.

Assignments:

1. Explain how and why the CHR\$() function has to be used in order to PRINT a statement with quotation marks.
2. Help Janet write a user-friendly INPUT statement. Write the code that will print the following:

ENTER A FIVE DIGIT NUMBER

☐ _ _ _ _ _

3. (The following is a continuation of Development 8.) Write a program that will open 26 sequential files and will store the file name in each file. This part of the program should be written using no more than five single-statement lines (not including an END statement). Then write another module, no longer than six single-statement program lines, that will obtain the file contents from the disk and print it on the screen.
4. Why are there different ASCII Code numbers for capital and lowercase letters?
5. Explain the process involved in programming a computer to alphabetize a list of words.

Assignment for More Advanced Students:

1. The directory of a disk listed three programs on it including one called TOWERS, one called PHRASE, and one called ASYLUM. A student typed the command to execute "towers" and the computer responded INVALID FILE NAME. The same message was printed when "phrase" or "asylum" was attempted, yet someone else ran the programs just before. Given that the proper load command was entered, what might be the cause of the trouble?

Topic 46 DATA SEARCHES

Performance Objectives: Students will be able to:

- . locate and print out a specific item in an array
- . find the highest and lowest number stored in an array

Vocabulary: New: search
Review: ASCII Code

Motivation:

- . Ten letters of the alphabet were randomly chosen by a person and INPUT into an array. How can we write a program that will check to see if an A was one of these letters?

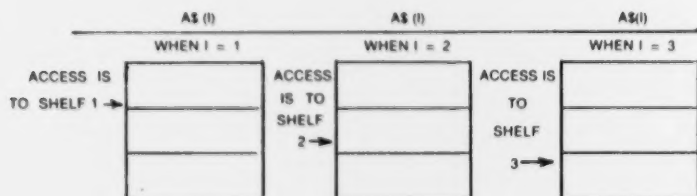
Aim: How can we find the largest and smallest elements in a series of numbers stored in an array?

Development:

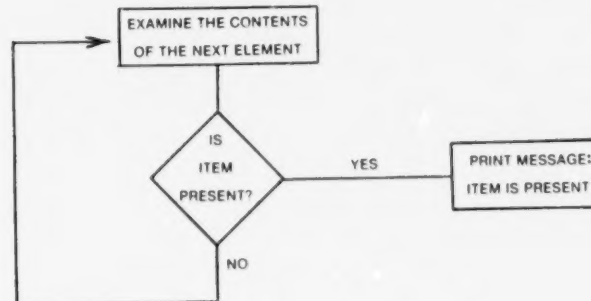
1. Refer to the Motivation and compare the problem to that of someone searching the shelves in a storeroom for a specific item.
 - . Elicit that in order to find a specific item all the shelves would have to be searched until the item was found. Explain that in order to find something in an array the same thing would have to be done.
2. Ask how we can examine the contents of the elements of an array and find a specific item.
 - . Guide the students to see that two things are involved: (1) we could use an IF/THEN statement to check the elements in the array and compare them to the item we are looking for; and (2) the FOR/NEXT loop is ideal for moving from element to element, as the loop utilizes an indexing counter.

Illustrate this by putting the following diagram on the board:

```
10 FOR I = 1 TO 3
20 IF A(I)=...
30 NEXT I
```



- . Elicit the following flowchart:



Have the students translate the flowchart into code that will solve the Motivation problem.

Acceptable answer:

```

10 REM SEARCH AN ALPHABET LOADED 10 ELEMENT ARRAY FOR THE LETTER "A".
20 FOR I = 1 TO 10
30 IF A$(I) = "A" THEN PRINT "A IS IN THE ARRAY"
40 NEXT I
50 PRINT "NO SUCH ITEM IS IN THE ARRAY"

```

- . Ask if the above program will work and elicit that an END statement is needed in line 30 to prevent both messages from being printed.
- . Ask how we could also ascertain which element contained the missing item.


```

30 IF A$(I) = "A" THEN PRINT "THE DESIRED ITEM IS ELEMENT ";I; "OF THE ARRAY"

```
- . Elicit that locating a number in an array would be performed in exactly the same way.
- 3. Refer to the Aim and ask how we could find the largest and smallest number in an array.
- . Elicit that the problem is very similar to that of locating an item in an array. Remind students that in the search for the letter A the statement we used was:


```

IF A$(I) = "A" THEN...

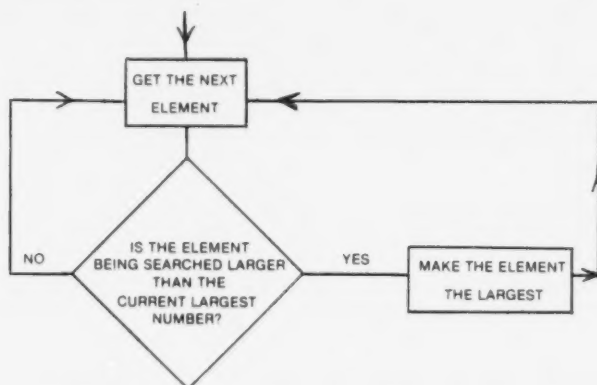
```
- . Define the word algorithm as a method of solving a problem and present the following algorithm for finding the largest number in an array of numbers.

Use the analogy of a teacher lining a class up in order of size. Review the process in which a child is arbitrarily chosen as the

standard for "tall" and other students are compared to him or her. If one is taller, then that student becomes the standard.

We could find the largest number by using the same reasoning. All we would have to do is to compare all the numbers in the array to the current largest number. If the first number in the array is greater than the current largest number (which is initialized at zero) then it in turn becomes the current largest and all numbers are now compared to it. In this way after the last number in the array is checked the current largest number will be the largest number in the array.

Display the following flowchart:



Have the students write the code for the flowchart. Suggest that the students assume that the numbers have already been INPUT.

Acceptable answer:

```

10 REM 10 ELEMENT ARRAY A(I); LG IS LARGEST #
20 REM THE NUMBERS HAVE ALREADY BEEN INPUT
30 FOR I = 1 TO 10
40 IF A(I) > LG THEN LG = A(I)
50 NEXT I
  
```

. Ask for the value of LG the first time line 30 is executed.

Stress that when we start we have no knowledge of what the largest number is. Guide the students to realize that the value we start with is of no consequence since the search will locate the largest. All we need for a starting number is one which is in the array. Let the first element in the array be the largest.

Add the assignment line to the program:

```

10 REM 10 ELEMENT ARRAY A(I); LG IS LARGEST #
20 REM THE NUMBERS HAVE ALREADY BEEN INPUTTED
25 LG = A(1)
30 FOR I = 2 TO 10
40 IF A(I) > LG THEN LG = A(I)
50 NEXT I
60 PRINT "THE LARGEST NUMBER IN THE ARRAY IS "; LG
  
```

- Have the students write a program that will identify the smallest number in an array of 10 numbers. Indicate that for the purposes of the lesson it is assumed that the numbers have already been INPUT.

Acceptable answer:

```
10 SM = A(1)
20 FOR I = 2 TO 10
30 IF A(I) < SM THEN SM = A(I)
40 NEXT I
50 PRINT "THE SMALLEST NUMBER IN THE ARRAY IS ";SM
```

- Ask the students if we could use this method to search an array and find the word closest to A or closest to Z in an array of words. Briefly describe the ASCII Code, reminding the class that letters are represented inside the computer by a number code, that this code is the same for all computers, and that it is called the ASCII Code.

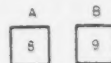
Indicate that the letter A has a lower code number than B does and that we could write the same type of IF/THEN comparison statement that we did with numbers:

IF A\$ > B\$...

Stress the fact that we are really comparing the code numbers in this statement and not the letters directly.

- Relate that we have just learned how to identify the largest and smallest members of a numerical array. Indicate that we will now learn how to exchange the positions of array members.

- Illustrate the case in which A = 8 and B = 9.



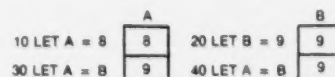
Ask how can we exchange the values of the two variables so that A will equal 9 and B will equal 8.

Elicit that we cannot simply exchange the values of the variables by writing:

A = B and B = A

because A = B gives A a value of 9. Writing, at this point, B = A merely reassigns a value of 9 to B.

Reinforce this concept by using the following diagram:



Elicit that we need a variable to hold the original value of A temporarily so that it does not get lost when A becomes 9.

Show the following code:

```
10 A = 8
20 B = 9
30 TEMP = A
40 A = B
```

Elicit that line 50 should read B = TEMP.

Trace the code using the following diagram:

PROGRAM LINE	VARIABLE VALUES AFTER EXECUTION OF PROGRAM LINE		
	A	B	TEMP
10 LET A = 8	8		
20 LET B = 9		9	
30 LET TEMP = A			8
40 LET B = A	9		
50 LET B = TEMP		8	
FINAL VALUES:	9	8	8

THE VALUE OF A HAS NOT BEEN LOST

Show a simple way to tell if the switch is written correctly. If the diagonal pattern shown here exists then the switch is correct.

```
TEMP = A
      A = B
      B = TEMP
```

Application:

1. Write a program that will INPUT the names and ages of five people, three of whom are 65 years of age or older. The data will be stored in two arrays. The program will print out the names of those people 65 and older.

Acceptable answer:

```
10 CLS : REM HOME or SHIFT (CLR/HOME) may be required
20 FOR I = 1 TO 5
30 INPUT "ENTER THE NAME"; A$(I)
40 INPUT "ENTER THE AGE"; B(I)
50 NEXT I
60 CLS : REM HOME or SHIFT (CLR/HOME) may be required
70 FOR I = 1 TO 5
80 IF B(I) >= 65 THEN PRINT A$(I) " IS "; B(I); " YEARS OLD"
90 NEXT I
```

Summary:

1. How is the IF/THEN statement used in a search routine?
2. How can a FOR/NEXT loop be helpful in searching through an array?
3. Explain the method of representing letters in the computer that makes it possible for us to select from a list of words the one alphabetically closest to the letter A. What is the purpose of the temporary variable used when exchanging the values of two variables?

Assignments:

1. The Lilliput Library has hired you to write a program so that the computer will be their card catalog.

Your program should be able to:

- . Use READ/DATA to input the titles of their entire (and magnificent) five-book collection.
 - . Store them in an array.
 - . Search the list for a specific title and indicate whether that title is in the collection. If the title is not there, it will state, "NO BOOK FOUND." If the title is in the collection, it will tell which position in the array it occupies.
2. Write a program that will INPUT a list of ten words and store them in an array. The program will then print the word that is alphabetically closest to A and the one alphabetically closest to Z.
 3. Write a program that will INPUT the last name, first name, address, city, state, and zip code of five people. Each item will be stored in its own array. Should someone enter a name, the computer will search for the person and print the person's last name, first name, and address if found. If the person is not found the computer will print a message saying there is no such person on file.

Assignment for More Advanced Students:

1. Write a program that will choose ten random numbers and put them into an array. The program will then print headings for three columns, labeling the first ELEMENT NUMBER, the second ODD, and the third EVEN. Each number in the array will be checked to see if it is odd or even and each will be printed in its appropriate column. The element number will be printed in its column. After the ODD and EVEN columns are printed, the largest and smallest numbers in each will be printed. Finally, the average of the even numbers and the average of the odd numbers will be printed.

Topic 47 LINEAR SEARCH

Performance Objectives: Students will be able to:

- . write a program to do a linear search for a numeric variable
- . write a program to utilize nested FOR/NEXT loops and MID\$ to perform a linear search for a phrase in an array of string variables

Vocabulary: New: linear search, key

Review: numeric array, string array, MID\$, FOR/NEXT, IF/THEN

Motivation:

- . Write a computer program to determine if any student received a final grade of 100 in a class with 200 students.
- . For what purpose might one need to find an element (key) in a numeric array?
- . For what purpose might one need to find a phrase of an element in an array of string variables?
- . What do we call the process of finding an element in an array? (searching)

Aim: How do we search for an element in a numeric or string array or for a phrase in an array of string variables?

Development:

1. We'd like a computer to search through a numeric array of credit card numbers to find a particular element (the key), ID number 235.
 - . The basic algorithm for a linear search is: each element of the array is compared successively with the key, beginning with the first element, until either an element in the array equals the key or the entire array is searched.
 - . To search through an array, we use a FOR/NEXT loop.
 - . If the key is found then a message is printed. A GOTO is then used to avoid the message after the FOR/NEXT loop which states that the key was not found.

```
10 DIM A(100)
20 DATA 131,278,...,829
30 READ A(I)
40 IF A(I)=235 THEN PRINT "ID number 235 is contained in this list":
   GOTO 60
50 NEXT I
60 PRINT "No one in this list has ID number 235"
70 END
```

2. Assume that each element of an array of string variables consists of a

student's name followed by grade, PASS or FAIL. Each element has 15 characters.

- . We'd now like the computer to search through the string arrays to determine if any student failed the course.
- . We'll need nested FOR/NEXT loops. The outer loop will vary the subscripts for each element of the array. The inner loop will be used with MID\$ to search through each element of the string array, which has a maximum of 255 characters, to determine if the word FAIL appears.

```
10 DIM A$(50)
20 DATA "RODRIGUEZ PASS ",,,,,,"ROSENRAUCH PASS"
30 FOR I=1 TO 50
40 READ A$(I)
50 FOR J=1 TO 12
60 IF MID$(A$(I),J,4)="FAIL" THEN PRINT "AT LEAST ONE STUDENT
   FAILED IN THIS CLASS": GOTO 100
70 NEXT J
80 NEXT I
90 PRINT "EVERYONE PASSED IN THIS CLASS"
100 END
```

Summary:

1. What is the basic algorithm for a linear search of a numeric array?
2. What is the purpose of the inner and outer loops in a linear search of an array of string variables to locate a phrase?

Assignments:

1. Write a program to determine if the digits 007 appear in an array of social security numbers.
2. Write a program to determine if the name REAGAN appears in an array of string variables of first and last names.
3. Determine the output of this program:

```
10 DATA CAREW .388,BRETT .390,WILLIAMS .406
20 FOR I=1 TO 3
30 READ NAME$(I)
40 FOR J=1 TO 13
50 IF MID$(NAME$(I),J,4)=".406" THEN PRINT NAME$(I):GOTO 90
60 NEXT J
70 NEXT I
80 PRINT "COBB .410      "
90 END
```

Assignment for More Advanced Students:

1. Write a program to determine which last name is most common in an array of 50 names.

Topic 48 BINARY SEARCH

Performance Objectives: Students will be able to:

- . write a program to do a binary search for a numeric variable
- . write a program to do a binary search for a string variable

Vocabulary: New: binary search

Review: numeric array, string variable, FOR/NEXT, IF/THEN, key, sorted

Motivation:

- . If an array is already sorted, must we search from the beginning to the end of the array in order to search for an element?
- . Why is it inefficient to do a linear search for a key in an array that is sorted?

Aim: Is there a faster method for searching for an element on a sorted list than a linear search?

Development:

1. Let us have the computer search through an array of string variables (50 book titles) to find a key (Gone with the Wind). Assume the array is sorted in ascending alphabetical order. In order to avoid linearly searching through each element, we will successively go to the middle and decide whether to look at elements above or below that point. This method is called binary search.
 - . The basic algorithm for a binary search of an array in descending order is as follows. First examine the middle element. If the key's value is less than that of the middle element, then the key is compared to the middle value of the first half. If the key's value is greater than that of the middle element, then the key's value is compared to the middle value of the second half. These steps are repeated until the key is found or until it is determined that the key is not present.
 - . A message is printed if the key is found, and a GOTO avoids the message after the FOR/NEXT loop that states the key was not found.
 - . The variable J is the beginning index of the interval under consideration, and K is the final index of the interval. The integer variable L is the middle index. If L is a title that is alphabetically before Gone with the Wind, then the new final index, the new K, is now one less than the old middle index, the old L. Therefore, the interval is from index J to the new K. If the old L is a title which is alphabetically after Gone with the Wind, then the new beginning index, the new J, is one more than the old middle index, the old L. Therefore, the interval is from the new J (which is the old L+1) to the original K. As long as the beginning index is less

than the final index, the computer goes back to step 90 to repeat the loop.

```
10 DIM A$(50)
20 DATA "ALL QUIET ON THE WESTERN FRONT","ZORRO"
30 FOR I=1 TO 50
40 READ A$(I)
50 NEXT I
60 J=1
70 K=50
80 L=INT((J+K)/2)
90 IF A$(L)= "GONE WITH THE WIND" THEN PRINT "GONE WITH THE WIND IS AN
    AVAILABLE TITLE":GOTO 150
100 IF A$(L)<"GONE WITH THE WIND" THEN K=L-1
110 IF A$(L)>"GONE WITH THE WIND" THEN J=L+1
120 IF J<K THEN 80
130 PRINT "GONE WITH THE WIND IS NOT AVAILABLE"
140 END
```

- . Note that the maximum number of times that a computer will search through a 50-list with a binary search is 5 times (for example, the 25th, then 12th, then 6th, then 3rd, and then 1st elements). In a linear search, a computer would need to search through the list 50 times.

2. Let us say that each element of a sorted (in ascending order) array of numeric variables consists of weekly salaries, from 100 to 10000 dollars. Write a program to do a binary search in order to determine if any employee has a salary of \$1000.

Applications:

1. Write a program to do a reverse alphabetical order binary search of a sorted array of names of cities to find if New York City is listed.
2. Consider a sorted array of times that a class meets (in descending numerical order). Have the user input a time that he or she is free. Write a program to have the computer determine if the class meets at the time that the user is free.

Summary:

1. What is the purpose of a binary, as opposed to linear, search?
2. What is the basic algorithm for a binary search?

Assignments:

1. Write a program to do a binary search of an array of serial numbers arranged in ascending order to determine if the serial number 10256 is present.
2. Write a program to do a binary search of an array of last names arranged in alphabetical order to determine if the name MONROE is listed.

3. For each of the above programs, determine the maximum number of times the computer will search through the list.

Assignments for More Advanced Students:

1. Write a program to do a binary search of a numeric array to count the number of times that a key is present.
2. Write a program to do a binary search of a string array to count the number of times that a key is present.

Topic 49

BUBBLE SORT: INTRODUCTION

Performance Objectives: Students will be able to:

- . describe the basic algorithm for a bubble sort
- . write the order of elements in a numeric array after each pass of a bubble sort
- . write the order of elements in a string array after each pass of a bubble sort

Vocabulary: New: bubble sort, pass

Review: numeric array, string array, IF/THEN, FOR/NEXT

Motivation:

- . Why might we want to arrange an array of names or any string variable in alphabetical order?
- . Why might we want to arrange numbers in ascending or descending order?
- . What do we call the method of arranging string variables in alphabetical order or numeric variables in numerical order (sorting)?

Aim: What is the basic algorithm for a sort of string and numeric arrays?

Development:

1. Let us consider an array of six numbers: A(1)=7, A(2)=9, A(3)=3, A(4)=35, A(5)=1, A(6)=42. We want to sort the numbers into ascending order.
 - . Each time we go through the complete array to compare elements is called a pass. On the first pass, the first two elements of the array will be compared and swapped if they are out of order. Then, still in the first pass, the second and third elements will be compared, and the third element will be swapped with the second if necessary. This process of comparing and swapping continues. Generally, we will need to have many passes through the array before the array is sorted. On each pass, the smaller elements "bubble" to the left. Hence, this method is called a bubble sort.
 - . Let us illustrate what happens to the order of the elements in the array after each pass. For example, in the first pass, the 7 and the 9 are not swapped, since the numbers are to be arranged in ascending order. However, the 9 and 3 are swapped, as are the 1 and 35.

Pass	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)

Start	7	9	3	35	1	42
1	7	3	9	1	35	42
2	3	7	1	9	35	42
3	3	1	7	9	35	42

Pass	A(1)	A(2)	A(3)	A(4)	A(5)	A(6)
4	1	3	7	9	35	42
5	1	3	7	9	35	42

Note that the array is now sorted into ascending order.

- Consider an array of string variables: A(1)="LATOUCHE", A(2)="ROSENRAUCH", A(3)="DIAZ", A(4)="STEIN", A(5)="ALLEN". We want to sort the array into alphabetical order.

Note that a computer arranges the ordering of letters so that a letter is considered to be less than another letter after it in the alphabet. For example, ALLEN<ROSENRAUCH, since A<R. If the first letter of both words match, the machine compares the next letter of each word, and so on. Write the order of the elements in this array after each pass of a bubble sort.

Applications:

- Write the order of the elements after each pass of a bubble sort to arrange a numeric array in descending order.
- Assume the values in a numeric array with 100 elements have already been read into a computer. Write a program segment to print the smallest value in the array. The basic algorithm is: let the variable, smallest, initially be the first element in the array. A FOR/NEXT loop will vary the index from 2 to 100.

An IF/THEN statement will compare each element of the array to the element designated "smallest." If the element is smaller than the "smallest," then that element will become the new "smallest."

Summary:

- Which elements of an array are compared during a pass in a bubble sort?
- What is the result after a pass of a bubble sort?

Assignments:

- Determine the number of passes needed in a bubble sort to arrange an array with eight elements in ascending order.
- Determine the number of passes needed in a bubble sort to arrange an array with "n" elements in ascending order.
- Write the order of the elements after each pass of a bubble sort to arrange a string array with seven elements in reverse alphabetical order.
- Write a program to print the largest element in a numeric array.
- Write a program to print the first element, alphabetically, of an array of ten strings.

Topic 50 **BUBBLE SORT: CONTINUED**

Performance Objectives: Students will be able to:

- . explain the basic algorithm for a bubble sort
- . write a bubble sort program to arrange a numeric array in ascending or descending order
- . write a program to determine the median of a numeric array

Vocabulary: New: pass, median

Review: numeric array, string array, IF/THEN, FOR/NEXT

Motivation:

- . Would the following two steps swap elements A(I) and A(J)?

A(I)=A(J)

A(J)=A(I)

If not, what additional step(s) would be needed?

- . Why might we want to know the middle value of an array of numbers?

Aim: What is the program for a bubble sort to arrange numeric or string arrays in ascending or descending order?

Development:

1. The basic algorithm for arranging elements in ascending order by a bubble sort involves an outer FOR/NEXT loop used for the index for each pass. The element with the lower value of the remaining elements in the array is determined on each pass. The index for the outer loop is compared to the remaining indices in the array, which are varied by the inner FOR/NEXT loop. For example, if the array has six elements, then the outer loop would be:

- . FOR I=1 TO 5 (since there are five passes). The inner loop is:
FOR J=I+1 TO 6 (for example, if I=2, then the second element must be compared from the third element, that is I+1, to the sixth.)

An IF/THEN statement is used to determine from a comparison of the A(I)th element to the A(J)th element whether a comparison is necessary. The following three steps below are needed for a swap of A(I) and A(J). TE is a temporary variable to hold the value of A(I) before A(J) is stored in A(I). Without TE the value of A(I) would be lost. A(J) then becomes equal to TE.

TE=A(I)

A(I)=A(J)

A(J)=TE

A bubble sort program for arranging six elements of a numeric array in ascending order is:

```

10 DIM A(6)
20 DATA 7,9,3,35,1,42
30 FOR K=1 TO 6
40 READ A(K)
50 NEXT K
60 FOR I=1 TO 5
70 FOR J=I+1 TO 6
80 IF A(I)<A(J) THEN GOTO 120
90 TE=A(I)
100 A(I)=A(J)
110 A(J)=TE
120 NEXT J
130 NEXT I
140 PRINT "The numbers in ascending order are:"
150 FOR L=1 TO 6
160 PRINT A(L)
170 NEXT L
180 END

```

Note that if the first element in each pass (A(I)) is less than an element with larger index (A(J)), then no swap is necessary. Therefore, the computer skips to step 120.

2. Write a bubble sort program to alphabetize ten book titles. Arranging a numeric array in ascending order is similar to arranging a string array in alphabetical order, since the inequality sign in statement 80 is < in both programs.
3. When a numeric array is sorted, the median is defined as the middle number. For example, if a sorted array has eleven elements, then the median is the value of the sixth element. If a sorted array has ten elements, then the median is the average of the values of the fifth and sixth elements.

Application:

1. Write a bubble sort program to arrange a numeric array with 12 elements in descending order. Write a bubble sort program to arrange an array with 12 addresses in reverse alphabetical order.

Summary:

1. What is the basic algorithm for a bubble sort to arrange elements in a numeric array in ascending order?
2. What is the purpose of the inner and outer FOR/NEXT loops in the bubble sort program?
3. What is the sequence of steps needed to swap elements in an array?

Assignments:

1. Write a program to print in descending order 20 numbers that were input by the user.

2. Write a program to alphabetically sort 15 movie titles that were input by the user.
3. Write a program to print the last name alphabetically of 10 names that were input by the user.
4. Write a program to print the median of a numeric array with 17 elements.
5. Write a program to print the median of a numeric array with 18 elements.

Assignment for More Advanced Students:

1. Input 12 last names with their first names. Let some of the last names, but not first names, be identical. Use a bubble sort to arrange the names in alphabetic order.

Topic 51

SELECTION SORT

Performance Objectives: Students will be able to:

- . describe the basic algorithm for a selection sort program to arrange numeric or string arrays in ascending or descending order
- . write the order of elements in a numeric or string array after each pass of a selection sort

Vocabulary: New: selection sort

Review: numeric array, string array, IF/THEN, FOR/NEXT

Motivation:

- . Why might we want a method other than bubble sort to sort arrays?
- . How can we arrange names alphabetically if some of the people have the same last name but different first names?

Aim: How do we write programs with a selection sort with string and numeric arrays?

Development:

1. Consider an array with six names. Let A\$(1)="RADIN", A\$(2)="SMITH", A\$(3)="RODRIGUEZ", A\$(4)="WILLIAMS", A\$(5)="ROSENRAUCH", A\$(6)="STERN". We want to sort the array into ascending alphabetical order.
2. Another method, other than a bubble sort, is called a selection sort. The basic algorithm for arranging the elements in ascending order, for example, involves a computer searching through an array to find the smallest value and exchange it with the first element. Then the computer searches through the array to find the second smallest value and exchanges it with the second element. These steps continue until the end of the array is reached. In each step of a selection sort, all items of the original array are considered in order to find the least item which is to be the next item of the final array.
 - . We will illustrate what happens to the order of the elements in the array after each pass. We will use the first two letters of each last name.

PASS	A\$(1)	A\$(2)	A\$(3)	A\$(4)	A\$(5)	A\$(6)
start	Jo	Sm	Ro	Wi	Br	St
1	Br	Sm	Ro	Wi	Jo	St
2	Br	Jo	Ro	Wi	Sm	St
3	Br	Jo	Ro	Wi	Sm	St
4	Br	Jo	Ro	Sm	Wi	St
5	Br	Jo	Ro	Sm	St	Wi

- . A program for selection sort is:


```

10 DATA "RADIN","SMITH","RODRIGUEZ","WILLIAMS","ROSENRAUCH","STERN"
20 FOR I=1 TO 6
30 READ A$(I)
40 NEXT I
50 REM THE OUTER LOOP GOES THROUGH THE ARRAY 5 TIMES
60 FOR I=1 TO 5
70 REM ASSUME A$(I) IS THE FIRST ALPHABETICALLY
80 FI$=A$(I)
90 REM LET J KEEP TRACK OF THE INDEX OF THE MINIMUM
100 J=I
110 REM COMPARE A$(I) WITH THE ELEMENTS BELOW IT TO FIND THE REAL
    MINIMUM
120 FOR I=I+1 TO 6
130 IF A$(K)>= FI$ THEN 160
140 FI$=A$(K)
150 J=K
160 NEXT K
170 REM SWAP A$(I) WITH MINIMUM
180 TE$=A$(I)
190 A$(I)=A$(J)
200 A$(J)=TE$
210 NEXT I
220 FOR L=1 TO 6
230 PRINT A$(L)
240 NEXT L
250 END

```

Applications:

1. Write a selection sort program to arrange a numeric array with ten elements in descending order.
2. Write a selection sort program to arrange an array with ten names of countries in reverse alphabetical order.
3. Write the order of the elements after each pass of a selection sort to arrange a numeric array in descending order.

Summary:

1. Which elements of an array are compared during each pass of a selection sort?
2. Write the selection sort basic algorithm.
3. During each pass of a selection sort, what is the sequence of steps to find the minimum (or maximum) value?
4. During a selection sort, what is the sequence of steps to swap elements?

Assignments:

1. Compare the speed of a bubble sort and a selection sort to sort 50 elements of a numerical array, 100 elements of a numerical array, and 200

elements of a numerical array. Does the size of the array affect the relative speeds of the two sorting methods?

2. Use a selection sort to print the median of a numeric array with 23 elements.

Assignment for More Advanced Students:

1. Input eight students' names with their high school overall averages. Write a selection sort program to print the students' names and respective averages in rank order, from the highest to lowest average.

Topic 52

EDITING A PROGRAM: LINE EDITING (RADIO SHACK, APPLE MBASIC, ETC.)

Performance Objectives: Students will be able to:

- . define "editor"
- . differentiate between a line editor and a full screen editor
- . use a line editor to make changes in a program line without retyping the entire line

Vocabulary: New: editor, line editor, screen editor

Motivation:

- . Assume you have written a swap routine that is one line long:

```
100 IF A>B THEN TEMP = A: A = B: B$ = TEMP
```

You type RUN and the computer flashes a message:

```
TYPE MISMATCH ERROR
```

You have inadvertently put a '\$' after the 'B' in 'B\$ = TEMP'.

Do you have to retype the entire line in order to correct this one small error?

Aim: How can we use a line editor to make changes in a defective program line?

Development:

1. Define a computer editor as a routine that will enable a programmer to make changes without having to retype the whole word or line.
- . Indicate that there are two types of error correcting routines in use in today's computers: screen editors and line editors. With a screen editor, corrections can be made to anything on the screen, be it a direct statement (PRINT "JOHN") or a program line (10 PRINT "JOHN"). All one has to do is to move the cursor to the location of the desired change and make it.

Elicit some examples of computers having full screen editors. Among them are the Apple, Commodore 64, PET, and IBM PC.

With a line editor, corrections can only be made to program lines that have been already entered. Entering a line editor generally requires a programmer to type in the word EDIT and the number of the line to be edited. For example,

```
EDIT 10
```

Remind the students that we are not talking about changing something

when they are first entering a line; in that case they can correct as they type.

Elicit some examples of computers having line editors. Some examples are the IBM PC, the APPLE II with MBASIC, and the TRS-80.

- . State that line editing will be studied in this lesson and screen editing in the next lesson.
- 2. Briefly show how corrections are made to a line as it is being entered into your machine.
 - . Elicit that if the person presses the carriage return key (ENTER), then the line is stored in memory as it was typed. To change it, the line editor will have to be used.
 - . Elicit that if the carriage return had not been pressed the person could move the cursor back to the error and make the change. Show that in moving the cursor over, all the letters up to the error are erased.
- 3. Remind the students that if they wish to change a whole program line that is already in memory all they have to do is retype it. The computer automatically replaces the old line with the new one.
 - . Elicit that in order to erase a whole line they merely have to type the line number and press ENTER. The computer replaces the old line with nothing (the new line), thus effectively erasing it.
- 4. Line editing of program lines:
 - . Review the concept that line editing allows one to make changes in program lines without having to retype the whole line; that it works with lines that have already been entered and are stored in RAM.

- . Getting into the edit mode (MBASIC):

There are two ways to get into the MBASIC edit mode: (1) typing EDIT and the line number followed by <CR> puts the computer into the edit mode; and (2) a SYNTAX error automatically places the computer in the edit mode:

```
? SN ERROR IN 30
READY
30
```

The 30 in the last line indicates that the computer is in the edit mode ready to edit line 30 where the SN error is.

- . Key letters in the MBASIC EDIT mode.

Explain that when in the edit mode, pressing certain keys causes special things to happen. Indicate that eight such keys will be described and that learning their functions enables one to use the edit mode.

State that using the edit mode saves lots of time and typing, but stress that until one learns the functions of those eight keys and becomes proficient in their use it might seem easier to retype the whole line rather than to edit it. Encourage the students not to retype.

Indicate that we will first list the functions of the eight edit keys and then distribute worksheets and examples for them to practice with. Reiterate that only with practice will they truly understand what the commands do and find out how to best use them.

MBASIC EDIT KEYS

PRESS -----	EFFECT -----
L	To list the LINE you are editing and keep you in the edit mode.
C	To enable you to CHANGE a character.
D	To DELETE a letter.
I	To INSERT things into a line.
X	To list the entire line and enable you to add things to the end of it. (Using this key makes the computer behave as if the person were typing the line for the first time; i.e., if the user moves the cursor to the left, characters are erased.)
Q	To cancel all changes and get the computer out of edit.
A	To cancel all changes and keep the computer in the edit mode.
H	To chop, or HACK, the line from the cursor's position to its end.
shift ↑	To get OUT of I, H, X modes of edit.
ENTER	To finalize all changes and get the computer out of edit.

5. Distribute the following worksheet and have the students work on their own.

WORKSHEET on LINE EDITING in MBASIC (page 1 of 7)

This worksheet will serve only as an introduction to the edit mode of MBASIC. There are valuable advanced features that will not be covered. After you are comfortable using the commands presented here, look in your manual for the others. What will not make sense now will then.

The following program will serve as our model for learning what the MBASIC edit mode is like. It will INTRODUCE you to the effects of the special keys. ONLY practice will make you proficient in their use.

```
10 PRUNT "THUS AT THEE ED ITT"  
20 PRIMPT "MOODE OF THHE TRS-80"  
30 IT IS ERY HPFLL  
40 FOR WECAN YUSE OT ANGE INKS WITTH OUT RETIPP ING THE LANE FOREVER  
   MORE.
```

Using the edit mode we shall change it to read:

```
10 PRINT "THIS IS THE EDIT"  
20 PRINT "MODE OF THE TRS-80"  
30 PRINT "IT IS VERY HELPFUL"  
40 PRINT "FOR WE CAN USE IT TO CHANGE THINGS WITHOUT RETYPING THE WHOLE  
   LINE"
```

Note: Line numbers cannot be changed by using the edit mode. The line must be retyped with the new number in order to do this.

1. Start by entering the practice program exactly as it is written, and then clear the screen.

- . Now edit line 10. First, type:

EDIT 10 (ENTER)

We see:

10

Now press the space bar a few times and notice that each time it is pressed another letter of the line appears.

Next, use the left arrow to move the cursor back to the beginning of the line.

Press any letter except the special ones we discussed (L, C, D, I, X, Q, A, H) and the letters S and K. Notice that none of the letters you press are added to the line.

Press the space bar and keep it pressed until the entire line is listed. Notice that the cursor does not go past the end of the line as originally entered.

WORKSHEET on LINE EDITING in MBASIC (page 2 of 7)

Press the left arrow key (←) a few times and notice that the letters disappear, that is, they are removed from the screen but they are still stored in memory.

After erasing about half the line press the ENTER key. Notice that the whole line is listed and that we are out of the edit mode. (How can you tell that this is so?)

Return to the edit mode to edit line 10. We do this by typing:

```
EDIT 10 (ENTER)
10
```

Type L to list line 10 and keep the computer in the edit mode. We see:

```
EDIT 10
10 PRUNT "THUS AT THEE ED ITT"
10
```

To change the U in PRUNT to an I, move the cursor over so that it is just below the U:

```
10 PRUNT "THUS AT THEE ED ITT"
10 PR
```

Press C (for Change) once and notice that nothing happens.

Now press I. Notice that an I has replaced the U.

To prove that the change has occurred, press L.

. Change THUS to THIS doing it the same way you changed PRUNT to PRINT.

Prove that the change was made.

Change AT to IS.

Prove in one of the following two ways that the change was made:

- Press L, which lists the line with the changes made and keeps you in edit.

or

- Press ENTER, which finalizes the changes and gets you out of the edit mode.

Press ENTER to finalize the changes.

Now DELETE the following:

WORKSHEET on LINE EDITING in MBASIC (page 3 of 7)

- an 'E' in 'THEE'
- the space in 'ED ITT'
- a 'T' in 'ED ITT'

. Return to the edit mode so that you can edit the rest of line 10.

List the line while in the edit mode.

Using the space bar, move the cursor to where the E to be deleted is:

```
EDIT 10
10 PRINT "THIS IS THEE ED ITT" (gotten by pressing the L key to list
    the line)
10 PRINT "THIS IS THE
```

Press the D key. We see:

```
10 PRINT "THIS IS THE!E!
```

The computer puts ! ! around the E to tell us that it has deleted what is inside them.

Note: On some versions of MBASIC slash marks (/ /) are used instead of exclamation points.

What methods could we use to see the changes? What is the difference between the two methods?

a-----

b-----

Press L twice, and notice what happens each time.

Caution: After using the delete function, don't backspace. We will see why shortly.

Now remove the space in ED ITT:

- Space over so that the cursor is right after the D:

```
10 PRINT "THIS IS THE ED
```

- Press the D key. You see:

```
10 PRINT "THIS IS THE ED! !
```

What does the ! ! tell us?

Remove the extra T from EDITT. You can remove either one, but try removing the first one.

WORKSHEET on LINE EDITING in MBASIC (page 4 of 7)

- Space over so that the cursor is under the space occupied by the T you wish to remove:

10 PRINT "THIS IS THE EDI

- Press D. (EDI!T!)
- Press L. What happens?-----
- Press L again. What happens?-----
- Are you still in the edit mode? How can you tell?

- To effect the change what would you do?-----
Now do that.

- . Now let's look at the problems created by backspacing after having used the D key. Tell the computer that you wish to edit line 10.

- Move the cursor to the beginning of the word THIS and delete the first three letters (THI)
- Now backspace until you can go no farther or until all the ! ! are removed.
- Press L twice. Notice that you have not succeeded in cancelling the deletions by backspacing. The letters are still missing.
- . Press A and notice the 10. This tells us that we are still in edit.
- List the line and note that pressing A cancelled all the changes made since you entered the edit mode but left you in the edit mode.
- Re-edit the line making all the changes again.

When finished, what do you press to finalize the changes?

2. So far we have seen how to change a character by using the C key and how to delete a character by using the D key.

- . Using these two keys change line 20 to read:

20 PRINT "MODE OF THE TRS-80"

(Remember: Type EDIT 20 ENTER)

3. Next we will edit line 30 (30 IT IS ERY HPFLL) changing it to:

30 PRINT "IT IS VERY HELPFUL"

WORKSHEET on LINE EDITING in MBASIC (page 5 of 7)

Looking at it we can see that there are no changes or deletions that we can make to do this. The corrections we have to make are to insert missing letters.

For this we have to use the I key.

- . Enter the EDIT mode and list the line:

```
30 IT IS ERY HPFLL
30
```

The first thing we have to do is insert the word PRINT, a space, and a quotation mark before the word IT.

Leave the cursor right after the 30.

- . Press I to enter the insert phase of the edit mode.

Now anything we type, including spaces, will be inserted in the space between 30 and IT.

Insert the characters PRINT ".

Now you need to move the cursor to the space between IS and ERY. You cannot do so unless you get out of the insert mode, since each time you press the space bar you will be adding a space to the line.

- . To exit the insert mode press the shift key first, and, keeping it pressed, press the up arrow. Then release them.

Space over to the proper place and insert a V to make ERY into VERY.

Go on and make HPFLL into HELPFULL. (Remember to get out of insert first.)

What would happen if, while in insert, you pressed the L key?

What would happen if you pressed A while in the insert mode?

- . Press the Q key. Notice that you have quit the edit mode. How can you tell this?

What you have not seen as yet is that, besides your getting out of edit, all changes made since you entered edit were cancelled.

Go back and re-edit the line.

4. You will now see how to change line 40 from:

```
40 WECAN YUSE OT ANGE INKS WITH OUT RETIPPING THE LANE FOREVER MORE
```

WORKSHEET on LINE EDITING in MBASIC (page 6 of 7)

to:

40 PRINT "FOR WE CAN USE IT TO CHANGE THINGS WITHOUT RETYPING THE WHOLE
LINE"

- . So far you have learned to use the following keys:

L C D A Q I shift ENTER

There are two other commands you should cover: X and H.

Tell the computer that you wish to edit line 40.

EDIT 40
40

Press the space bar and keep it pressed until the cursor gets to the end of the line and won't go any further. Remember that pressing any letter other than one of the special letters will have no effect. Try it.

Now press L to list the line and try the same thing. Again, no effect.

- Press X. Note that the entire line was listed (as with L). However, unlike the situation with L you can now make any changes you want to. For example, you can erase letters by using the left arrow, and you can add letters.

Now use the ← to erase the words FOREVER MORE and change the word LANE to LINE. Don't press any other keys after you make these changes.

You now wish to list the line. The L key should be used for this. Try it. Notice that instead of listing the line the letter L is added to the line. This is similar to what happened in the insert mode.

Erase the L.

If you were finished editing all you would have to do to effect the changes would be to press what? -----.

However, you are not finished. You get out of the X mode the same way you got out of the insert mode. How was that?-----.
Do it.

Now press L.

X is very useful for making changes at the end of the line or for adding new things on to the end of a previous line.

WORKSHEET on LINE EDITING in MBASIC (page 7 of 7)

- There is another way to remove FOREVER MORE and that is by hacking it, or chopping it off.

Press Q to cancel the changes you made and return to the edit mode.

Press the space bar, keeping it pressed down until the cursor is after the E of LANE and then press H.

This hacks off the words FOREVER MORE and allows you to add anything you want to the end of the line.

List the line to see what happened.

Now press Q to cancel the changes and quit the edit mode.

Re-edit line 40 changing it to the desired version.

- . Try editing a line and changing its line number. Can you do this?

Assignments:

1. What is the purpose of an editor?
2. What is the difference between a line editor and a full screen editor?
3. What functions do the following keys have in the MBASIC edit mode?
C, D, I, L, H, Q, A
4. Would you use X to remove a word that was found at the beginning of a line? Explain.
5. A student was writing a quiz program and had entered the following when she accidentally pressed the carriage return.

23 IF A\$="JACK" OR A\$="JIM" OR A\$="JOHN" .

What is the best edit command for her to use so that the words "THEN PRINT "CORRECT" could be added?

6. When in the edit mode, pressing either the L key or the ENTER key lists the line. When would you use the L key and when would you use the ENTER key?
7. How do you get out of the insert mode? What other modes can be exited this way?
8. You type the following line: 10 PRNT "GO MAN GO!". You enter edit, press I and insert an I making the word PRINT. There are no more corrections to make. Do you have to press Q before you press the ENTER key?
9. Using the line editor is not always the best method of changing text. You wish to erase line 30. Which is the easiest way to do it:

Enter the edit mode to edit line 30, press X and backspace to erase all the characters on the line.

Type 30 and <ENTER>.

Topic 53

EDITING A PROGRAM: FULL SCREEN EDITING (COMMODORE)

Note: For Apple computers, change the control keys to the acceptable ones.

Performance Objectives: Students will be able to:

- . define what "editor"
- . differentiate between a line editor and a full screen editor
- . use a screen editor to make changes in a program

Vocabulary: Review: editor, line editor, screen editor

Motivation:

- . Assume you have written a swap routine that is one line long:

```
100 IF A>B THEN TEMP = A: A = B: B$ = TEMP
```

You type RUN and the computer flashes a message:

```
TYPE MISMATCH ERROR
```

You have inadvertently put a '\$' after the 'B' in 'B\$ = TEMP'.

Do you have to retype the entire line in order to correct this one small error?

Aim: How can we use a screen editor to make changes in a program?

Development:

1. Have the students explain what a computer editor is.
 - . Have them differentiate between the line editor and the full screen editor. (Using the line editor one can only edit a line already stored in memory while using the full screen editor one can change anything on the screen or in the memory, direct statement, or program line).
 - . Remind the students that the last lesson dealt with use of a line editor and that this lesson will deal with the use of a full screen editor.
2. Have the students name the Commodore 64 (include the PET and VIC 20) and the Apple as those computers having full screen editors.
 - . State that while the basic operations of full screen editors are the same no matter which computer is being used the specifics are different. The following will deal specifically with the Commodore 64 editor.
 - . Indicate that the lesson will consist of a brief lecture and a lab in which students will complete a worksheet.

3. List the following keys as being used in editing on the Commodore 64:



- . Elicit that pressing any of the keys gives the bottom function while first pressing the shift key, keeping it pressed, and then pressing any of the other keys gives the top function.

- . Elicit the functions of all the keys.
- . Explain that the above keys can be used in two different ways: for immediate effect and for later execution.
 - For immediate effect, press crsr and the cursor moves.
 - For later execution, the keys can be used as a cursor control code placed in a string whose action will happen only when the program is run.

State that the computer advises the programmer that it is inserting the cursor action into the string by placing a code in the string.

Illustrate this by placing the following codes on the board as an explanation:

If you press	The computer prints
-----	-----
HOME	S
CLR	
INST	
DEL	(DELETE is the only control key which works inside an open quote print statement. Later in the lesson the students will learn a method of placing a code for DELETE in a string.)

- . State that the above codes will be seen when the students do the worksheet.
- . Explain to the students that changes made on the screen are not changed in memory. Thus they are not made permanent until RETURN is pressed.

WORKSHEET on SCREEN EDITING for the COMMODORE 64 (page 1 of 4)

1. Clear the screen by pressing the clear/home key.
2. Type the words LEAPING LOUIE and do not press RETURN. (LEAPING LOUIE).
3. Note the effects of the following:

- . Press CRSR twice → →
- . Press CRSR four times ↑↑↑↑
- . Press CRSR twice ← ←
- . Press CRSR twice ↓ ↓
- . Press HOME

4. DELETE. Move the cursor to the L of LOUIE.

- . Press DEL. Notice that the space to the left of the cursor was deleted. (LEAPINGLOUIE)
- . Delete the ING of LEAPING. (LEAPLOUIE)

5. INSERT. In insert, spaces are inserted first and then characters are put in these spaces.

- . To insert the ING back into LEAPLOUIE and to move the cursor past the two words, first leave the cursor on the L of LOUIE. Then press INST four times. (LEAP LOUIE)
- . Notice that: (1) the cursor stayed where it was; and (2) spaces were inserted pushing the word LOUIE over.
- . Press the ING keys. (LEAPING LOUIE)

Note the location of the cursor between the two words.

At this point all that is left is to move the cursor past the two words (LEAPING LOUIE). There are two possibilities. For the first, notice what appears to be a space between the words where the cursor is. Four spaces were inserted for characters and only three were entered ING). The space between the words awaits a character, which in this case happens to be a space.

- . Press the SPACE BAR to enter the space.

Use the → to move the cursor past the word LOUIE. Note that the screen now shows the following: LEAPING LOUIE.

WORKSHEET on SCREEN EDITING for the COMMODORE 64 (page 2 of 4)

In order to see the second possibility, return the situation that existed before the space was inserted: (LEAPING LOUIE)

Delete the ING by moving the cursor to the L of LOUIE and deleting four characters. Now insert four spaces, placing ING in three of them. The situation is now as it was before the space was entered via the SPACE BAR.

- . Instead of pressing the SPACE BAR use the CRSR to move the cursor over to the right of LOUIE.

Notice that you see the following:

LEAPING LOUIE

The " " between the two words was put there by the computer because you inserted spaces for four characters and only put in three (ING).

When you pressed the " " the computer thought that the fourth character you wanted to insert was the control character for " ". It complied with what it saw as your wishes and told you so by printing the " ". Since that took up the fourth space you inserted the cursor key, and action now returned to normal and the cursor moved to the end of the word LOUIE.

Why didn't the computer print the " " when you press the SPACE BAR in the first case?

6. You have now seen the effect of all the keys involved with editing. The next step is to edit a PRINT statement.

- . Clear the screen. Now start with a mathematical statement. Enter the following and do not press RETURN after entering it.

PRINT 92 = 8 + 2

Use only the crsr keys (not the DELETE key) to move the cursor to the = and replace it with a * making the statement PRINT 92 * 8 + 2. Press RETURN to have the statement executed.

- . Try doing the same operation, this time with the same expression as a string. Enter the following and do not press RETURN after entering it.

PRINT "92 = 8 + 2"

Make sure that you close the quotes. Use the CRSR keys (not the DELETE key) to change the statement to:

PRINT "92 * 8 + 2.

Then press RETURN to execute the statement.

WORKSHEET on SCREEN EDITING for the COMMODORE 64 (page 3 of 4)

Note that the change was easily effected in both cases.

- . Retype the string statement but do not close the quotes and do not press RETURN:

```
PRINT "92 = 8 + 2
```

Try using the CRSR to change the = to *.

With the exception of the DEL key, pressing any of the edit keys (CLR, HOME, INST, ← → ↑ ↓) gives you the symbol that was demonstrated in lecture.

Note that you are typing inside a set of open quote marks and the computer thinks that you wish the cursor movements to be part of your string. It tells you that it has placed them there by printing the symbols that you see.

- . Press the DEL key. See how it works.

In order to effect the change, you can do one of two things:

- Delete up to the =, change it to a * and retype the rest of the statement.
- Close the quotes. The computer interprets this to mean that you have finished your string and that anything done to it from now on will be in the nature of editing. It therefore sets all edit keys to work normally.

Try both methods.

- . There is a way to include the DEL key function in a string.

Type PRINT "92 = 8" and don't press RETURN.

Move the cursor to the =. (PRINT "92 = 8")

Insert a space (PRINT "92 = 8")

Press DEL. Notice that the computer printed a T to tell us that it will execute the delete function when it executes the PRINT statement. (PRINT "92T = 8")

Press RETURN and note the result. (9 = 8) The 2 was deleted by the embedded delete function.

7. So far, you have edited on screen statements. To edit a program line that is not on the screen, first list the line. Then move the cursor up to the line and edit it.

- . Clear the screen and enter the following:

```
10 PRINT "2 + 3 ="  
20 PRINT 6
```

- . Clear the screen again.

WORKSHEET on SCREEN EDITING for the COMMODORE 64 (page 4 of 4)

- . Change line 10 to read

```
PRINT "3 + 3 ="
```

- . Type LIST 10 and press RETURN (10 PRINT "2 + 3="). Move the cursor to the 2 and make the change. Do not press RETURN. Then use the key to move the cursor down to the next blank line. Now list the program.

Note that the change was not made. Even though a change is made on the screen, the change in memory is not made until RETURN is pressed.

Go back and change the 2 to a 3, this time making the change permanent.

8. There is one other invaluable feature. It is the ability to change line numbers and to duplicate lines at the same time.

Type NEW.

Clear the screen.

Type 10 PRINT 22 and press RETURN.

Move the cursor up to the 1 in 10 and press 2 changing the 10 to a 20.

Press RETURN.

Repeat the process twice more, changing the 20 to 30 and the 30 to 40 pressing RETURN each time.

List the program and note that you have four duplicate lines.

```
10 PRINT 22
20 PRINT 22
30 PRINT 22
40 PRINT 22
```

Should you wish to erase the original line, just type the line number and press RETURN.

- . Can you suggest some ways in which the above would be useful?

Assignments:

1. Explain the difference between a line editor and a screen editor.
2. Justin noticed that he had just typed PRINT "SHE SELLS SHESHELLS" and wished to change it to PRINT "SHE SELLS SEASHELLS". He pressed the crsr key to get back to the word SHESHELLS and got a cursor in the correct place on the screen. Why did he get this, and what should be done to correct the original error?
3. Duplicate the following line as line 100 and line 200 by using the full screen editor.

10 PRINT "HO!"
4. Explain how the screen editor could be used to change a line number.

Topic 54

DOCUMENT PREPARATION

Performance Objectives: Students will be able to:

- . load a word processing program
- . name a document
- . enter text within a document
- . save a document on a disk
- . add additional text to an existing document

Vocabulary: New: document, main menu

Review: save, load, cursor, files, store, retrieve, text entry, edit

Materials:

- . word processing program
- . documentation for the program
- . sample text to be entered and stored in the word processor

Motivation:

- . Distribute Handout #1 to the class. Present these questions:

Ms. Clark's secretary must type the letter in the handout to a customer. The secretary has the option of using a typewriter or a computer with a word processing program. If you were the secretary, what would you do? Why?

Aim: How do we use a word processing program to enter and save text?

Development:

1. What is word processing? (Distribute Handout #2.)

What does a word processing program do?

Who uses a word processing program?

How might a word processing program be helpful?

What are the advantages of a word processing program over a typewriter?

2. Now, let us become Ms. Clark's secretary. We will type the letter using a word processor. How is this accomplished?

- . Demonstrate how to load the word processing program into the computer.

- . Introduce the main menu if your processor uses one. The main menu is the starting point for all word processing activities on some machines. It appears each time the word processing program is loaded and permits the user to accomplish a specific task.

- . Depending upon the word processing program used, the main menu will permit the user to enter text, edit text already stored within the program, access the printer, and accomplish other program-specific tasks. Illustrate these tasks by reviewing the main menu for the specific word processing program being used.)
3. Instruct the students in opening a document for the first time (if your processor requires it).
 - . Opening a document is the way the user instructs the computer to work with a specific document.
 - . We will open our document by naming it "letter." (Give specific instructions, based on your word processing program.)
 4. Now, we will enter the text itself. Type in the body of the letter. Do not type in the inside address, the salutation or complimentary close.
 - . Explain the basic procedures in entering text. Illustrate the fact that the text will automatically "wrap around" at the end of the line without pressing the return key.
 - . Illustrate the use of the backspace and all other cursor move keys.

Note: Certain other text entry features are machine specific and should be discussed at this time, as explained within the specific documentation for the word processing program.

5. At this point, discuss the specific cues of the particular word processing program that indicate how much memory has been used, how much memory remains, and any other information that may be displayed on the screen as text is entered.
6. Now, let us save the text we have already typed in. How may that be accomplished? What might we do to complete this task? (Give a command to the computer to save the existing text.)
 - . At this time identify the specific command that must be given to the computer to save the document.

- . After that has been done, have the class return to the main menu

We now have a document called "letter" saved within our word processing program.

Have we finished creating the letter? (no)

What have we not done? (We have not entered the address, salutation, or complimentary close.)

How might we recall the text we have entered so that we can complete the task, entering the additional information?

- . Give instructions to the class as to how text may be retrieved for addition of more information.

7. Have the class enter the address and salutation at this point.
 - . After this is completed, have the students save this additional information on disk.
 - . Now have the students add the complimentary close to the letter to complete their task. Then, have the students save the final document.

Applications:

1. Assemble in groups to create a continuing story. One student begins by typing in a sentence. The second adds a sentence. Continue the process until all students have typed sentences.
2. Enter your last writing assignment from English class into the word processing program. Store the document for later printing.

Summary:

1. What are the advantages of word processing programs?
2. How do we use word processing programs?

Assignments:

1. Create a word processing history book. Research the history of the development of word processing. Information may be obtained from computer manufacturers and software companies.
2. Create a glossary of word processing commands used in text entry.
3. Research the many different word processing programs available. Identify any specific characteristics unique to a particular word processing program. Determine their prices and list their good features and bad features.
4. Create a class newspaper with all text stored on a word processor for publication later through use of a printer.

HANDOUT # 1

April 28, 1984

Mr. Brian Larson
2934 Everett Way
Yonkers, New York 13304

Dear Mr. Larson:

Enclosed please find the copy of A GUIDE TO WORD PROCESSING, which you requested.

As you review this textbook, I'm sure you will be immediately aware of the value of a text written especially for your word processing program. The author, Mr. Elvin, has been teaching word processing at The Everyday School for more than fifteen years. His knowledge of computers and, most importantly, word processing is clearly reflected in his book.

As you will note, our texts are distributed by Summer's Press. Orders may be placed by writing to them at: 14 Empire Drive, Yonkers, New York 13304, ATTN: Lois Clark. Please note also that there is an educational discount of 25% on any order.

We thank you for your interest and hope that you will enjoy this text.

Sincerely,

Lois Clark
Sales Representative

HANDOUT # 2

What is a word processing program?

A word processor is a computer program that makes it easier to type, revise, store, retrieve, and print text than with a typewriter. Word processing is a way of preparing professional-looking documents.

When you use a word processing program on your computer and you are typing at the keyboard, whatever you type appears on the screen and not on a piece of paper. At the same time, what you type is being stored in the computer's memory. (This area is called the workspace.) Because your text is being stored in the computer's memory, you can move forward or backward through the text to read what you have written and to revise the material. You can use the editing functions of the program to make changes in the text while it is in the memory.

In order for you to be able to use your material again, you must save it on disk or cassette. When you turn the machine off you erase what is in the computer's memory, or workspace.

Your material can be reloaded at any time into the memory of the computer so you can continue to work on it or have your word processing program print out a copy for you. Remember, you must first load your word processing program into the computer before you can continue to work on your material.

Topic 55 **EDITING TEXT ON A WORD PROCESSOR**

Performance Objectives: Students will be able to:

- . identify the editing function of a word processing program
- . insert text
- . delete text
- . save a revised document
- . use the various editing commands for a specific word processing program

Vocabulary: New: overtyping, tabs

Review: save, load, delete, insert, edit, cursor, files, revise, store

Materials:

- . word processing program
- . software specific documentation for the word processing program
- . chart of editing commands for the word processing program being used

Motivation:

- . Load the word processing program and retrieve the document called "letter" which was prepared in the last lesson.
- . Ms. Clark's secretary forgot an additional paragraph which should be located immediately before the last paragraph. Must the secretary retype the entire letter over? How might a word processor be used to correct this mistake?

Aim: How can the word processing program be used to edit an existing document?

Development:

1. Distribute Handout #3 to the class. Locate the additional text which must be added.
2. Review the steps required to reach the main menu.

Review the main menu and identify the specific procedure required to reach the EDIT mode. Proceed to the EDIT mode.

3. We wish to insert additional text within the letter. What must we first do? (Identify the location where the additional text must be entered and move the cursor to that position.)

Instruct the class on how the cursor can be moved to a specific location. Practice by moving the cursor to specific locations within the document.

Move the cursor to the position where the additional text must be entered.

4. Give specific instructions as to how the INSERT function is accessed.

Identify the characteristics of the screen that indicate that the program is in the INSERT mode. (Instruct the students in regard to the specific INSERT functions of the word processing program in use. Include examples of insertion of text, carriage returns, blank spaces, and lines.)

- . Practice accessing the INSERT function.
 - . Now, once the INSERT function is accessed, have the class insert the additional text to be entered.
 - . When that is completed, have the students save this material within the document.
 - . Have each student choose another area of text to insert some further information. Insert that information into the body and save.
5. Review the existing text. Ms. Clark would like a copy of the same letter to be sent to Ms. Ellim. Must the entire letter be rewritten?
- . How can we modify the letter so that it can be used for Ms. Ellim? (Delete the original salutation and inside address and in their place insert "Dear Ms. Ellim:" and her address.)
 - . How can we delete the existing salutation?
 - . Review the specific DELETE instructions for the word processing program in use.
 - . DELETE the existing salutation.
 - . How can we now write Dear Ms. Ellim? (We can now insert the salutation, using the procedures for insertion of text.)
 - . After inserting the new salutation, have the students review the body of the text for any spelling mistakes they might have made. Have them delete the mistakes and insert the correct word(s).
 - . Review the specific DELETE instructions for the word processing program now being used. (Specific instructions for deleting letter by letter, as well as whole words, sentences, and paragraphs should be explained.)
6. Explain how OVERTYPE can be used within a program as an alternative to the DELETE and INSERT functions. Often an error in typing occurs because the user hits the wrong key. In such a case, the easiest way to correct this mistake is to use the OVERTYPE (or CHANGE) function and just retype the correctly spelled word over the mistake. This is an alternative to deleting the entire word and then inserting the correctly spelled word.
- . With OVERTYPE, a single function is used instead of two functions to correct an error.
 - . Practice OVERTYPE within the text of the letter.

7. Examine any other EDIT functions in the word processing program being used.
 - . Identify the TAB function and show how it can be used. (Show how it can be used for the the complimentary close of the letter.)

Application:

1. Create a document and have the students access it. Give instructions as to what should be inserted and deleted and tell the students to correct all spelling errors. Then, have the students save the corrected document.

Summary:

1. How are the DELETE and INSERT functions used within a word processing program?
2. How can the OVERTYPE function be used as an alternative to the DELETE and INSERT functions.

Assignments:

1. Create a document and save it. Review this document and decide what you would like inserted and deleted. Give your document to the person to your right, and give a series of instructions for insertions and deletions. Now, follow the directions and correct the document you have been given. Correct all spelling errors.
2. Make a table of all EDIT function commands, specifying what each does.
3. Examine two different word processing programs. Compare and contrast the EDIT functions of each.

HANDOUT #3

(to be inserted at the end of the second paragraph, in the
body of the letter in Handout #2)

This material is clearly explained. Your students will feel comfortable
reading this easy-to-understand text.

Topic 56

THE SEARCH/FIND FUNCTION ON A WORD PROCESSOR

Performance Objectives: Students will be able to:

- . find and replace characters or words
- . define a block of text
- . move a block of text
- . identify and use multifunction commands in the word processing program being used

Vocabulary: New: block, block moves, find, search, replace, multifunction
Review: insert, delete, overtype

Materials:

- . word processing program
- . program documentation
- . chart of text
- . editing commands for the word processing program being used

Motivation:

- . Distribute Handout #4A. Have the students read the selection, enter the document, save it, and edit it if there are any mistakes.
- . Distribute Handout #4B. Compare it to handout #4A. Have the students indicate on Handout #4A any differences that exist between the two.

Aim: How can we modify this document by moving the existing text?

Development:

1. Review all modifications that must be made. How can we perform all of the modifications? Must we retype the whole paragraph using the INSERT function and then the DELETE function to eliminate it from its present location?
 - . A block is a group of characters. It is any number of characters, whether text or codes. It can be as small as a single line or as large as several paragraphs or pages.
 - . Once you define a block you can move, copy, or delete it.
2. When you want to quickly locate a specific word or place within a document, you can instruct the computer to find that position within the document. It is not necessary for you to have to read the entire text to get to the specific place.
 - . The FIND function permits you to define a group of characters, a string, and then locate that string within the document.
 - . Using the specific directions for your word processing program instruct the class in using the FIND function.

- . Practice using the FIND function to locate specific strings within the document.
 - 3. Now, let us use the FIND function to locate the position in our document we are looking for.
 - . We are now at the point where we would like to eliminate the text in this position and instead put it further down within the document. How can we do this? (Use the DELETE function to remove it from the existing position and then use the INSERT function to place it in the new position.)
 - . Instead of using DELETE and INSERT, we can move this block of material using the MOVE function.
 - . Instruct the students on how to do BLOCK MOVES with your word processing program.
- Note: Certain word processing programs will permit block moves with automatic deletion of the text in the original position as well as moves with no effect on text in the original position. Give the students the specific instructions needed to perform BLOCK MOVES with deletion and without (usually called move and copy).
- . Now, using this technique, complete the first BLOCK MOVE, deleting the text from its original location.
 - . Have the class complete the additional moves within the document.
 - 4. Certain word processing programs permit the user to create his or her own command. These multifunction commands are sometimes helpful for accomplishing your task quickly.
 - . If your machine permits multifunction commands, instruct the students in the use of the multifunction commands of the specific word processing program being used.

Application:

1. List in random order the steps required to change a flat tire. Using the word processor, record the steps in random order within a document entitled "tire". Then, using BLOCK MOVES, rearrange the steps in sequential order.

Summary:

1. How can the BLOCK MOVE function be used as an alternative to the DELETE and INSERT functions?

Assignments:

1. Create a glossary of terms and instructions that indicate the multifunction capabilities of your word processing program.
2. Write an invitation to a party. Using BLOCK MOVES, move specific items of the text of the invitation to different places within the document.

HANDOUT #4A
THE PRINTER

Advantages:

Dot Matrix Printer:

The dot matrix printer is less expensive than the daisy wheel printer and is capable of doing graphics.

Daisy Wheel Printer:

The daisy wheel printer will create print that is letter quality and can be used for important correspondence. As with certain typewriters, the user can change the type fonts for different print styles and sizes.

Care and Feeding:

1. Check that the printer has paper and that the ribbon is in good condition. There must be no obstacle to interfere with the flow of paper.
2. The ribbon should be checked periodically to make sure that it hasn't stretched or come out of its holder and that it is still sufficiently inked.
3. Paper should be kept in the printer at all times, and the printer should be kept covered when not in use.

How to Use:

1. There are two popular types of printers, dot matrix and daisy wheel. The dot matrix printer has a printing head consisting of a number of wires, which pop out and make dots on the paper in the form of letters, numbers, and other characters.

The daisy wheel printer types one full character after another as a typewriter does.

2. A printer is capable of producing hard copy, or printout, of program listings and other data.

Every character (number, letter, etc.) travels from the computer to the printer through a cable and interface. Signals travel in the form of a numerical code called the American Standard Code for Information Interchange (ASCII).

Disadvantages:

Daisy Wheel Printers:

The daisy wheel printer is usually slower and noisier than the dot matrix printer. It is also more expensive to buy.

Dot Matrix Printer:

Since these printers do their work with wires that come out and strike the paper, the letters are made up of dots. The density of the dots depends on the printer: The more dots per letter, the nicer the print is. Some computer-phobes do not like receiving correspondence produced on a dot matrix printer, no matter how dense the dots.

HANDOUT #4B
THE PRINTER

How to Use:

1. A printer is capable of producing hard copy, or printout, of program listings and other data.

Every character (number, letter, etc.) travels from the computer to the printer through a cable and interface. Signals travel in the form of a numerical code called the American Standard Code for Information Interchange (ASCII).

2. There are two popular types of printers, dot matrix and daisy wheel. The dot matrix printer has a printing head consisting of a number of wires, which pop out and make dots on the paper in the form of letters, numbers, and other characters.

The daisy wheel printer types one full character after another as a typewriter does.

Advantages:

Dot matrix printer:

The dot matrix printer is less expensive than the daisy wheel printer and is capable of doing graphics.

Daisy wheel printer:

The daisy wheel printer will create print that is letter quality and can be used for important correspondence. As with certain typewriters, the user can change the type fonts for different print styles and sizes.

Disadvantages:

Dot Matrix Printer:

Since these printers do their work with wires that come out and strike the paper, the letters are made up of dots. The density of the dots depends on the printer: The more dots per letter, the nicer the print is. Some computer-phobes do not like receiving correspondence produced on a dot matrix printer, no matter how dense the dots.

Daisy Wheel Printers:

The daisy wheel printer is usually slower and noisier than the dot matrix printer. It is also more expensive to buy.

Care and Feeding:

1. Check that the printer has paper and that the ribbon is in good condition. There must be no obstacle to interfere with the flow of paper.
2. The ribbon should be checked periodically to make sure that it hasn't stretched or come out of its holder and that it is still sufficiently inked.
3. Paper should be kept in the printer at all times, and the printer should be kept covered when not in use.

Topic 57 PRINTERS

Performance Objectives: Students will be able to:

- . prepare a document for printing
- . access the printer for printing a document
- . compare and contrast a daisy wheel and dot matrix printer
- . recognize the capabilities of various printers to print in specific styles

Vocabulary: New: boldface, letter-quality printing, daisy wheel printer, dot matrix printer

Review: SAVE, LOAD, PRINT, LLIST, LPRINT, menu

Materials:

- . word processing program
- . revised documentation
- . chart of commands for the word processing program being used
- . printer
- . documentation for the printer

Motivation:

- . Type in the material on Handout #5, save it, and edit it if necessary.
- . How can we edit text within a document?

Aim: How can we now print the document?

Development:

1. Review the procedures for creating and editing a document.
 - . If we wish to obtain a hard copy of the document, what type of equipment is needed? (a printer)
 - . What is the purpose of using a printer? (obtain hard copy of specific text, programs, etc.)
2. Review the LLIST and LPRINT commands. If your machines utilize ports (PRINT# / or slots PR#1) review those commands.
 - . LIST program lines, stored in memory, to the printer.
 - . PRINT data on the printer.
3. Instruct the class in the procedures required to access a printer while using a word processing program. Have the class practice accessing a printer.
4. Generally a series of instructions must be given once a printer is

accessed. A menu will be illustrated giving specific instructions for certain features the word processing program will accomplish.

- . Identify the specific parameters that exist within the printing menu once the printer is accessed. Illustrate each option by permitting the class to use it.
5. Once the printer has been accessed and all print options have been tested, the printer is now ready to print text, according to your directions.
- . Instruct the class in the procedures that must be followed in order to print the entire text. Have the students then print out the text.
6. Identify the procedure used to halt printing and how it may be resumed.
- . How might we instruct the printer to print only a specific portion of text?
 - . What sort of instructions are needed in order to complete such a task? (The printer must be told where to begin printing and where to end printing.)
 - . Identify the specific instructions used within your word processing program that permit printing of sections of text.
 - . Have the students select a section of text they wish to print out and then proceed to print only that particular section.
7. What are the two types of printers that are most common? (daisy wheel printers, dot matrix printers)
- . How are they similar?
 - . How are they different?
 - . What are some advantages of each?
- Note: Handout #4B from the preceding lesson can serve as a review sheet on printers and may be referred to at this point in the lesson.
- . Identify some of the specific qualities of the class printer.
8. Instruct the students in underlining, creation of boldface, changing the number of characters per inch or lines per inch, establishing letter-quality printing, and any other specific printer qualities built into your class machine.
- . Have the class practice these options by modifying the existing text and printing.

Applications:

1. Prepare a list of brands of daisy wheel and dot matrix printers.

2. Make a sample of each type of printing style available using different printers.

Summary:

1. How can a printer be used to create hard copy of documents stored in a word processor?
2. Compare and contrast a daisy wheel printer and a dot matrix printer.

Assignments:

1. Create a poster of the printer menu used with your word processing program. Include within the poster an explanation of each entry.
2. Identify the different brands and models of dot matrix printers. List the print qualities of each, the speed at which printing can be done, specific unique characteristics of each, and the cost.
3. Create an advertisement for a printer showing many of its capabilities. Store the document in the word processor and print it out as a flyer.

Handout #5

CHILDREN'S ART

PRESENTING: A unique opportunity to display or sell your artwork

NO FEE

WHERE: Community City Hall, 125 Lafleur Ave.

RULES:

- * The theme will be "MY NEIGHBORHOOD"
- * All work must be original
- * Artist must be 16 years old or younger
- * Artist must be present on Saturday, May 6, for judging
- * Size of work must be no larger than 12 x 18 inches (plus frame if needed)
- * All art must be delivered to the Community City Hall no later than 1 PM Tuesday, May 2, and must be picked up by June 20. (We will not be responsible for any work remaining after June 20.)

PRIZES:

First prize: \$100 savings bond
Second prize: \$25 gift certificate redeemable at the ART LOBBY
Third prize: \$10 gift certificate redeemable at the ART LOBBY

Certificates of participation will be awarded to all who enter

Fill in the form below and attach it to the BACK OF YOUR ARTWORK. If you wish to sell your artwork, place ONLY the price tag on front of you work.

NAME

ADDRESS

SCHOOL

AGE

GRADE

Is your work for sale?

If so, what is the price?

Topic 58

DOCUMENT MANAGEMENT

Performance Objectives: Students will be able to:

- . explain how documents are combined
- . combine complete documents and portions of a document
- . rename documents
- . create documents and files
- . merge documents and files

Materials:

- . word processing program
- . documentation for the word processing program

Vocabulary: New: data files, merge, copy, merge document, field name record
Review: menu, document, text

Motivation:

- . Create a new document entitled "poetry", and type the text of Handout #6A into this document.
- . Save the document and print it.
- . Now, open a new document entitled "poem", and type the text of Handout #6B into this document.
- . Save the document and print it.
- . Since these two poems appear to be similiar, consolidate them into a single document.

Aim: How are documents merged?

Development:

1. Determine which of the two documents you would like merged into the other.
 - . Would anything be changed?
 - . How might this task be accomplished?
2. Using your specific word processing program, instruct the students in the procedures for inserting one document into the other.
 - . Practice this process using the two documents we now have.
 - . Suppose that only a portion of one document is to be merged into the other. How might this be accomplished?

- . Identify a section of the document "poem" to be merged into poetry".
 - . Would we use the same procedures that were used to merge the entire document "poetry" into the document "poem"?
 - . Have the students merge the specific portion of the document "poem" into the document "poetry".
 - . Choose a portion of the document "poetry" and merge it into the document "poem".
3. Since we now have material from the documents "poem" and "poetry" combined within one document, how can we eliminate the original document involved with the merger?
- . Why might we wish to eliminate the original document? (to save disk space)
 - . Review the need to conserve disk space when using a word processing program.
 - . Show how the original document "poem" can be eliminated by using the proper word processing command for the program.
4. Now, since we have only the document "poetry" left, we could rename this document "COLEPOEM".
- . Would it be necessary to retype the new document and name it?
 - . Would it be necessary to create a new document and then merge the two documents into this new one, or could we just rename the existing document?
 - . Instruct the students on the proper procedures for renaming a document.
 - . Create a new document called "letter" and type Handout #7 within this document.
 - . Save "letter" and print it.
 - . If we wished to use this letter for many people, what data would vary from person to person? (name, address, city, state, zip code)

Note: Some word processors cannot merge text with files and if yours is not capable of such merging instruct students to edit the file repeatedly by changing the necessary information over and over again. Tell them however, that there are word processing programs that have this facility.

6. Now create a MERGE DOCUMENT. A MERGE DOCUMENT is a document in which specific information is inserted at designated points from different documents. The specific information is inserted during printing.
- . Our MERGE DOCUMENT will use the original letter and, in the course of the printing, will insert the specific information needed for each individual, in order to create a personal letter for each individual.

- . Where on the handout is specific information to be inserted?
 - . What are the variables that exist within the letter?
7. Each portion of information is called a field name. Thus, name, address, city, state, zip code are each field names. Instruct the class on the specific instructions used to indicate a field name.
 8. Return to the document. Have the class revise the document to create a MERGE DOCUMENT.
 - . Save the document.
 - . Now, a merge document has been made by entering field names within the document. The system will replace the field names with the variables during printing.
 - . How do we create our variables?
 - . Develop a list of the name, address, city, state, and zip code of four friends.
 - . This list is going to be our data file to be used with our merge document. Once a merge document is created, the data file containing the variables to be inserted, must also be created.
 9. Instruct students on the procedures for creating a data file based on the instructions supplied with the specific word processing program being used.
 - . A data file is always entered as a nondocument file.
 - . A data file is made up of records. A record is one complete set of variables needed for each document. As a result, a record for our data file would contain the name, address, city, state, and zip code of an individual.
 - . Have the students create the data file.
 - . Now that the merge document and the data file have been created, they can be merged. When does this occur? (It occurs during the printing of the document.)
 10. Give instructions on the procedures needed to print the document, and have the class print the document

Application:

1. Create an invitation to an "End of the School Year Party" setting it up as a MERGE DOCUMENT if your word processor can merge. Otherwise print separate letters, editing as you go along. Then, set up the data files, using members of the class as the variables. Print the invitations.

Summary:

1. How can a document and data file be merged?
2. How can a document be altered?

Assignments:

1. Investigate a word processing program different from the one you have been using. Determine how document mergers can be done. Are the procedures the same as the ones you have been using? Create a chart of the differences and similarities.
2. Write a story, leaving portions of the story open for field names. Develop a list of words to be used as variables for the data files. Merge the documents and print the results.
3. Mr. Jones, the owner of a sporting goods store, would like to send a letter to those customers whose orders have been delivered to the store. He wants to inform them that equipment they ordered is now in stock. He would like to specifically state what equipment each person ordered and what the total bill is. Create the document and data file for this task. Then print the merged document.

HANDOUT #6A

In Xanadu did Kubla Khan
A stately pleasure-dome decree:
Where Alph, the sacred river, ran
Through caverns measureless to man
Down to a sunless sea.

HANDOUT #6B

So twice five miles of fertile ground
With walls and towers were girdled round:
And there were gardens bright with sinuous rills,
Where blossomed many an incense-bearing tree;
And here were forests ancient as the hills,
Enfolding sunny spots of greenery.

HANDOUT #7

January 27, 1984

Mr. Derick Fram
99 Elton Place
Brooklyn, New York 11205

Dear Mr. Fram:

The annual meeting of the stockholders of the Everready Pump Corporation will be held on February 21, 1984 in our corporate headquarters building in Wilton, Connecticut.

Please make every effort to attend this meeting, because several important issues be discussed.

Sincerely,

David B. Lamb
Everready Pump Corporation

Topic 59
UNDERSTANDING TEXT STORAGE
BY EXPLORING SEVERAL WAYS OF CREATING ASCII FILES

Performance Objectives: Students will be able to:

- . reevaluate past procedures that they have used to create text files
- . identify situations in which it is more expedient to create files without the use of a word processor
- . write BASIC programs that will create text files, read text files, and append text files to each other
- . save files on diskette and write menu-driven programs that will allow access to those files
- . write BASIC neatly formatted programs to create and save neatly formatted daily financial records as text files

Vocabulary: New: concatenate, ASCII FILES

Review: read, write, close, open, position, append, files,
sequential files

Motivation:

- . My accountant recently remodeled his office. In addition to new wallpaper, a couch occupied a wall where two huge steel file cabinets used to stand. On his secretary's desk I spied a new microcomputer. What type of change probably occurred in that office?
- . What are some of the ways we have of creating files on a computer?
- . What are some of the limitations of creating files on a word processor?
- . When would it be useful to save the output of a program as a text file?
- . What kinds of information are generally saved in files? Is there any advantage to not using software to save this information?
- . What is a menu-driven program?
- . Can programs be written that can create many different types of files?
- . Can we use the same program to read a list of names as we use to read a recipe?

Aim: What kinds of information are generally saved in files, and what is the best method for creating each one of these files?

Development:

1. Develop a list of the types of information that can be saved as a file.
 - . Discuss the advantages of using the computer to create these files.
 - . Review with students the different methods that they have already used to create text files.

- . Have students see that address files are best created by a word processor, student records benefit from the power of an electronic worksheet, sequential files are excellent manipulative tools for storing huge lists of data such as a large company's inventory, and random access files are valuable when time saving is a major consideration.
2. Elicit the limitations of using a word processor to create and manipulate text files. List the limitations on the chalkboard.

- . Include sorting and arithmetic operations.
- . By way of review have students write a file-create program. Use this program to input the names of the students in the class:

Suggested file-create program:

```

10 REM FILE-CREATE PROGRAM
20 C = 1
25 INPUT "PLEASE ENTER A NAME. ";A$
30 A$(C) = A$
40 C = C + 1
50 INPUT "MORE NAMES?";T$
60 IF LEFT$(T$,1) = "Y" THEN 25
70 INPUT "NAME THE FILE";F$

```

Note: Now open a file for output according to your machine's unique language.

```

120 FOR X = 1 TO C - 1
140 PRINT A$(X)
150 NEXT X

```

Note: Now close the file.

```

170 END

```

3. After creating the name file, catalog the disk. Show the students that the file that was created is a text or ASCII file.
- . Have students understand the necessity of writing a program to read this file.
 - . Write a general program that can read any file.
 - . Have students use the program to read their name file.
 - . Inform students that so far they have not used the power of the computer to their advantage.
 - . Have them suggest an alphabetical listing but leave this for an Assignment.

4. Tell students that they are now ready to create text files from within a BASIC program.
 - . Pose the following problem: The student union has offered a prize to the senior class that sells the most Sing tickets. These tickets sell for \$4.65 to Student Union members and \$6.25 to nonmembers. We are in charge of keeping the daily sales records for each of the eighteen senior classes. We will use our computer to save each day's record of sales as a text file.
 - . Review formatting techniques and user-friendly programs.
 - . Review dollars-and-cents formatting and alignment of decimal points.
 - . Introduce the term concatenate and produce six-character figures with the decimal point as the third character from the right.
 - . Review arrays, dimension statements, and string manipulations.
5. After creating several daily files, have the students write a correctly formatted Read program to read their files.
 - . Have the students recall several days' files and observe the listing of the files they saved.
6. Review the term menu-driven.
 - . Tell students that a recipe file can be created and saved using this technique. Furthermore the Read file can be written to adjust the ingredients in each recipe to the desired amount.
 - . Recipes can be saved using the file-create program written at the beginning of this lesson. Each recipe can be saved by name. For example:

"Chocolate cake", "roast turkey"
 - . A neatly formatted listing of these recipes should be developed. It could be saved as a file and then sorted in alphabetical order and brought to the screen.

Applications:

1. Rewrite the original File Read program to include a section that sorts alphabetically. Rename this file appropriately to indicate its use and save it on your disk.
2. Catalog your disks and use DELETE, KILL, SCRATCH, or ERASE to eliminate any unwanted files we created today. What is the purpose of doing this?
3. Tell whether each of the following is true or false.

It is always better to write files with a word processor. (T or F)

Text files are run the same way that BASIC programs are run. (T or F)

Arithmetic operations cannot be performed on string variables. (T or F)

Numeric variables can be changed to string variables using the STR\$ command. (T or F)

There is no BASIC command that can change an alphanumeric string to a numerical variable. (T or F)

Files can never be renamed. (T or F)

Numbers can never be included in file names. (T or F)

ONERR-GOTO commands are useful error-trapping techniques. (T or F)

Sometimes it is expedient to GOTO the step that includes the GOTO command. (T or F)

Summary:

1. When is it expedient to write files with a word processor?
2. Are there situations when we need routines that word processors cannot handle?
3. Once a file-create program is written, is it ever necessary to write a new program to open a new file?
4. What is the purpose of a Read file program?
5. What are menu-driven programs?
6. What is the advantage of setting up data files in a single string and putting more than one data field into it?
7. What are some of the uses for concatenating numbers and strings?
8. Is it ever possible to enter a file directly from the keyboard?

Assignments:

1. Recall two days' files that were created with the Sing program.
2. Create a third file that will merge the other two. Include a section that combines the daily totals for each class.
3. Write a program to create a credit file for a small business. Include a 5-digit customer number, a 25-character customer name, and a credit coding of your choice (such as A, B, C, D, E). Have sections of your program sort by each field.
4. Assume you have been assigned to create the following ASCII files. Next

to each file write the method you would use to create the file, such as word processor, electronic spreadsheet, BASIC program.

- (a) Student's cumulative school record
- (b) Biology Department library
- (c) Record of monthly rainfall in New York City, 1964-1984
- (d) Patient history at Mercy Hospital, 1980-1982
- (e) Record of household expenses for Mr. Ramirez
- (f) Daily sales at Sam's Bargain Store
- (g) Mr. Gold's Stock and Bond Portfolio account
- (h) Amy's family's and friends' birthdays

Topic 60

USING A WORD PROCESSOR TO MODIFY BASIC PROGRAMS

Performance Objectives: Students will be able to:

- . format BASIC programs so that they can be read by a word processor
- . use the BASIC statement EXEC to change a file created by a word processor back to a BASIC program (Apple only)
- . use a word processor to merge programs
- . create a file of subroutines that can be used to facilitate the writing of lengthy programs
- . use the word processor to facilitate changing numeric variables to string variables
- . make use of the word processor's ability to find a specific word when changing variable names in a BASIC program
- . draw pictures using the word processor that can be changed into a BASIC program

Vocabulary: New: text, EXEC, find word, renumber, replace character
Review: word processor, OPEN, CLOSE, READ, WRITE, merge, subroutine

Motivation:

- . What do we have to do to a BASIC program that arranges numbers in numerical order to change it to one that alphabetizes lists of data?
- . What problems do we encounter when we try to edit our program?
- . What power does a word processor have that would greatly enhance our ability to edit a program?
- . Is there a way of transforming our program into a text file that can be read by a word processor?
- . Can these files, once created, be changed back to BASIC programs?
- . Are there any restrictions that we must be alert to while writing these programs on a word processor?

Aim: How can we use the word processor to modify BASIC programs?

Development:

1. Have students understand the many advantages of using the power of the word processor to modify their programs.
 - . Develop a list of such advantages.
 - . Elicit problems that occur when editing a program, and show how the processor can help. The following table contains suggestions:

BASIC editing problems

Word processor advantage

Difficult to insert words	
and symbols.....	Easy to insert
Rearranging line numbers.....	Easy to modify
Unable to merge.....	Easy to merge
Tedious to find a variable	
name.....	Easy to find word
Tedious to rewrite the same	
expression.....	Easy to create Macros
Unable to test appearance of	
formatted screen.....	Easy to print
Difficult to create pictures	
with print statements.....	Easy character count
Unable to change variable	
name.....	Easy to replace character
Time-consuming to delete	
non sequential steps.....	Easy to delete
Hard to rearrange entire	
sections.....	Easy to copy lines

2. On most machines there is a Save command that allows a BASIC program to be saved as a text or ASCII file. Refer to your operator's manual for the correct syntax for your machine. APPLE users will need the following:

```
10 POKE 33,33
20 D$ = CHR$(4)
30 PRINT D$; "OPEN PROGRAMNAME.TEXT"
40 PRINT D$; "WRITE PROGRAMNAME.TEXT"
50 LIST 1,63999
60 PRINT D$;"CLOSE":TEXT
70 END
```

Stress the purpose of changing column width. Explain the addition of the word "text" to the program name so as not to have two programs with the same name. Review the procedure of writing files and explain the LIST statement.

Have students write a short program to sort numbers. Load this program into the processor with the above addenda.

3. Caution students to write only in capital letters. Have them explain why this is necessary.

Elicit the need to limit their program lines to 255 characters. Discuss how this can be done.

Stress the appropriate use of the carriage return.

Teach the use of the EXEC statement to translate the file you want to process back to BASIC. Tell the students that the EXEC statement directs the computer to accept the text file from the diskette on which it is saved as though it were normal keyboard input. The computer thinks it is receiving information from the keyboard when, in fact, it is reading a

text file from their diskette. Exit the word processor and write:

EXEC NUMERICAL SORT.TEXT

When completed name NUMERICAL SORT and save to the diskette. Point out to students the series of prompts that appears on the screen as the EXEC command takes over their computer. Explain that they were unable to see the file as it was rewritten. Tell them that if they wish to watch this procedure they can do so with the MON C,I,O. With an advanced group, or in a situation where time is not a factor, the teacher might want to introduce this command.

Since all information sent between the computer and the diskette is not normally displayed on the monitor, the MON command is a useful tool. It enables the user to see all or part of this procedure. The parameters C, I, and O specify disk Commands, computer Input, and computer Output, respectively. At least one of these parameters must be present, and no rule as to their order exists. This command will remain in effect until it is turned off with NOMON C,I,O.

4. Have students test the new program to see that it is in fact a new BASIC program that has been created from their numerical sort.
- . Suggest the creation of a subroutine that finds all proper divisors of a number. This is a useful subroutine that can be used when finding prime, perfect, abundant, and deficient numbers.

Create a short program beginning at line 500.

Suggested program:

```
500 REM SUBROUTINE BEGINS HERE
505 SN = 1
510 FOR D = 2 TO SQR (N)
520 IF N / D = INT (N / D) THEN 540
530 GOTO 560
540 IF D = N / D THEN SN = SN + D: GOTO 570
550 SN = SN + D + N / D
560 NEXT D
570 RETURN
580 REM SUBROUTINE ENDS HERE
```

Add the following lines to complete the desired program.

```
10 REM AMICABLE NUMBERS
15 HOME : PRINT
20 INPUT "PLEASE ENTER A NUMBER ";A
30 N = A
40 GOSUB 500
50 HOME
60 PRINT : PRINT : PRINT "THE POSSIBLE MATE FOR "A" IS "SN
65 PRINT : PRINT
70 B = SN
80 N = SN
```

```

90 GOSUB 500
100 IF A = SN THEN PRINT A" AND "B" ARE AMICABLE NUMBERS": GOTO 120
110 PRINT A" AND "B" ARE NOT AMICABLE NUMBERS."
120 END

```

- Have students save this program, rename it, and use the EXEC command to change it into a BASIC program. (Apple users only.) As a further exercise the numbers can be saved in arrays. This program can then be merged with the numerical sort program by using the merge capability of the word processor. The programs can be quickly renumbered and the variables adjusted to the same arrays.

5. Have students write a program to draw these two figures:

XXXXXXX	XXXXXXX
X X	X X
X X	X WATCH X
X STOP! X	X X
X X	X OUT X
X X	X X
XXXXXXX	XXXXXXX

- After the program has been written the class can return to the original figure drawn and change the word STOP to the expression WATCH OUT.
- Other road signs can be created that are useful when writing user-friendly programs.

6. Show students how a processor finds and replaces characters.

- Suppose a program has been written for the PET and we now wish to run it on the VIC 20. Every occurrence of the word PET has to be changed to VIC 20. Suggestion:

```

10 PRINT "GOOD MORNING I AM MR.PET"
20 INPUT "WHAT IS YOUR NAME";A$
30 PRINT "HELLO ";A$;" MR. PET IS PLEASED TO MEET YOU"
40 PRINT "DO YOU KNOW HOW TO USE THE PET COMPUTER?"
   etc.

```

- Introduce the FIND AND REPLACE command. Have students see how the cursor stops at every occurrence of PET and replaces it with VIC 20.
- Explain how the cursor is made to move forward or backward depending on the formatting of the command. Elicit other uses for this command.

Applications:

1. Write a program that will find the area of a rectangle given the coordinates of its vertices in order. Load the program into the processor and edit it to find the area of a triangle.
2. Discuss the commands that can be used to lighten the chore.

3. Enter the word processor and draw the picture:

```
*****
*                               *
* PROGRAM LOADING *
*                               *
*****
```

4. Code this program in BASIC and save it. Use the EXEC command (Apple only).
5. Save this as a useful subroutine.
6. Give students a routine that draws an axis on the high resolution screen. Have them write a program that draws the equation

$Y = -X^2$

at the center of the screen.

Delete the first six lines, add the appropriate REM statements, use EXEC (Apple only), and save as a BASIC program.

Summary:

1. What would happen if we loaded a program into the processor that was not previously prepared with the addenda?
2. What advantages are there to using a processor to edit a program?
3. Why is it beneficial to let the processor find characters for us?
4. Why is it necessary for us to keep renaming our program?
5. Is there any advantage to saving the program text that we create?
6. What other ways can be used to merge programs with subroutines?
7. Why do we bother to renumber on the processor when we have a renumber program?
8. Are the processor's tabs useful when editing?
9. Can we ever use lower case letters in our BASIC programs?
10. What other way is there to insert words in PRINT statements? Why don't we use that method?
11. What is the easiest way to rearrange program lines?
12. In your opinion, what is the major benefit of editing with a word processor?

Assignments:

1. A nice way to begin all programs is with the following routine.

```
10  REM *****
20  REM *
30  REM * PROGRAM NAME *
40  REM * WRITTEN BY *
50  REM * YOURNAME *
60  REM * DATE *
70  REM *
80  REM *****
```

Create this subroutine, add the proper addenda, and save it as a text file with your other subroutines. It can then be merged with other programs.

2. Write a program that will graph any straight line by simply inputting the slope and the y intercept. Merge this program with axes saved previously.
- 3 Load the numerical sort program that you wrote into the word processor.
4. Use the REPLACE CHARACTER command to find and change the variable name to B(I).

Topic 61

USING A WORD PROCESSOR TO MODIFY BASIC TEXT FILES

Performance Objectives: Students will be able to:

- . use a word processor to edit BASIC text files
- . maintain these files
- . use the power of the word processor to merge files
- . reformat existing files to use as address files that can be merged with letters and reports
- . make several new files out of an existing one

Vocabulary: New: dataset, data base, file maintenance, address file
Review: merge, report, edit files

Motivation:

- . Is it ever necessary to change data in a file that we have created?
- . What is meant by the expression "editing a file"?
- . What are some of the ways to edit our files?
- . Is it possible to use one file to print mailing labels and merge with a letter or report?
- . What are some of the ways that we have merged files in the past?
- . Is it possible to examine a text file without a BASIC program?
- . Today we are going to look at a new way of reformatting files. We will change data, add and delete fields, merge, segment, and do general file maintenance with the aid of a word processor.

Aim: How does a word processor aid us in file maintenance?

Development:

1. Explain that all text files created by BASIC programs, can be easily loaded into a word processor. Caution students to be aware that we are not referring to binary or command files or ASCII files created at the keyboard. We will deal only with sequential or random access files.
2. Have students boot their word processing programs and see how easily they can load their text files. Use a simple data base of students' names.
3. Delete several datasets by erasing the line, and insert others with the insert command.
4. Add phone numbers to already existing names and addresses.

5. Make sure the input and output file have the same name so that students can read their newly reformatted file.
6. Develop a list of types of files that can be easily maintained using this procedure. Some examples follow.

File

Student records
Inventory file
Airline schedules
Financial statements
Major league sports
Investment portfolio
Retail store record of purchases
Credit cards issued

File maintenance

Add test grades
Update number of items on hand
Change prices
Add charges, adjust credits
Update player performance, scores
Add purchases, delete sales
Add field "Returns"
Change expiration date

7. Give students experience in merging files. Create a new file from two already existing files. Then pose this problem: A large company keeps customer records departmentalized. Sometimes the same customer name occurs on more than one file. For a general mailing it is convenient to merge the contents of the customer list files from both the mail order division and the credit card department. We want our files merged and alphabetized and all duplicates eliminated.

Step 1. Name the output file "General Mailing".

Step 2. Load the file named "mailorder" into the word processor by using it as your input file.

ALAN	100234
BAKER	100367
COHEN	100435
COLMAN	100537
DEAN	100642
FRIED	100768
GALLAGER	100878
LOO	100998
MENDEZ	100999

Step 3. Position the cursor at the end of the mailorder file and use the Merge command to access the credit card file. If your credit card file looked like the following,

ARCHER	100156
BAKER	100367
KARP	100441
LOO	100998
NATEY	101322
RIOS	102788
SMITH	103245
TSANG	105377
YOUNG	109876

then the merged result would be:

ALAN	100234
BAKER	100367
COHEN	100435
COLMAN	100537
DEAN	100642
FRIED	100768
GALLAGER	100878
LOO	100998
MENDEZ	100999
ARCHER	100156
BAKER	100367
KARP	100441
LOO	100998
NATEY	101322
RIOS	102788
SMITH	103245
TSANG	105377
YOUNG	109876

Point out that there are two duplicates, Loo and Baker. Do not delete the duplicates at this time. Point out the need to realphabetize. Since most word processors do not have the ability to alphabetize, it's a good idea to make use of the alphabetical sort program written in an earlier lesson.

Step 4. Save the new file called "General Mailing".

Step 5. Write a sort program.

Step 6. Use this program to sort alphabetically and to rename the file "GENMAIL"

Step 7. Reboot the processor and load the GENMAIL file. Now the duplicates are found next to each other and can be easily compared as to name and number.

. Delete the duplicates by deleting the line, and save the file.

. Discuss with students the value of using this procedure with long lists of names.

8. Teach students the method of changing an existing file to an address file for the purpose of merging with a report. Then pose this problem: A mail order firm keeps customer records in an alphabetical file. The information in this file is then used to issue monthly statements. The original file has been set up by use of an electronic worksheet. The procedure for reformatting and merging this file is as follows:

Step 1. The output file is going to be an address file. Name it STATADDR.

Step 2. Load the file by its original name.

NAME LAST	NAME FIRST	ACCOUNT NUMBER	PREVIOUS BALANCE	PAYMENTS RECEIVED	DEBITS ACCRUED	NEW BALANCE
ADAMS	FRED	A045	75.00	75.00	120.00	120.00
BROWN	ELLEN	B163	0.00	0.00	72.00	72.00
DUNN	LINDA	D312	245.00	200.00	144.42	189.42
FOO	SALLY	F009	114.00	114.00	458.25	458.25
GIGLIANO	VITO	G866	86.00	86.00	59.00	59.00
LOPEZ	TINA	L051	172.00	100.00	0.00	72.00
MANN	JUDY	M333	288.26	300.00	309.61	297.87
WHITE	VONDA	W701	35.14	35.00	25.00	25.14

Step 3. Delete the titles and reformat so that each item is on a separate line.

- . End each dataset with an end of address character if your machine requires one. A slash (/) is a good choice. Make sure you remember the position of each item.

Reformatted file:

```

ADAMS
FRED
A045
75.00
75.00
120.00
120.00
/
BROWN
ELLEN
B163
0.00
0.00
72.00
72.00
/
DUNN
LINDA
D312
245.00
200.00
144.42
189.42
/
FOO
SALLY
F009
114.00
114.00
458.25
458.25
/

```

GIGLIANO
VITO
G866
86.00
86.00
59.00
59.00
/
LOPEZ
TINA
L051
172.00
100.00
0.00
72.00
/
MANN
JUDY
M333
288.26
300.00
309.61
297.87
/
WHITE
VONDA
W701
35.14
35.00
25.00
25.14
/

Step 4. Create a new file to be used as the customer's bill.

- . Use the same name for the input file that you use for the output file.
- . When writing the bill refer to each line in the dataset by number. For example #1 refers to the first line, which contains the last name and #3 would be the account number. How would you indicate the new balance? (last line, #7)

Suggested Bill:

FRIENDLY MAIL ORDER COMPANY
NEW YORK CITY

MONTHLY STATEMENT

APRIL 1984

NAME: #1 #2

ACCOUNT NUMBER:#3

PREVIOUS BALANCE:#4

PAYMENTS RECEIVED:#5

DEBITS ACCRUED:#6

NEW BALANCE: #7

Kindly remit on or before May 15, 1984.

Thank you for your payment.

Step 5. Use the power of the word processor to merge the letter with the address file just created if your word processor can merge a letter with a previously created file.

9. Pose this problem: For a midyear report I usually send letters of commendation to my students whose averages are 97 or above. At the same time I send warning letters to those students whose averages fall below 67. Test marks and averages are kept on a file that I name for each class. Is there a way that I can use the word processor for this task?

Elicit the following procedure even if your word processor cannot merge.

1. Name the output file "Commendations".
2. Use the class list for the input file.
3. Delete all datasets where the average is below 97.
4. Save to be merged with a commendation letter.
5. Create a new output file named "Warning".
6. Use the class list as the input file.
7. Delete all datasets with averages above 67.
8. Save to be merged with a warning letter.

Applications:

1. Reformat the Commendation and Warning files as address files.
2. Write the letter of commendation and the warning letter. Make provisions to list all test grades as well as the current average in each letter. Merge letters with the address files.
3. Create a file of several books that can be found in a math department library. Do the same for a science department library. Include several duplications such as The Life of Einstein and The Second Law. Merge these files with the word processor. Use your sort program to alphabetize them and rename the merged file. Reload into the word processor and eliminate the duplicates.
4. Load the Customer Account file into the word processor. Change the five-digit account number to a seven-digit one and add a column for the zip code.

5. Use the word processor to create an inventory file for a stationery store. Give every item a six-digit product number. Sort this file with a numerical sort program and save under a new name.

Summary:

1. How does the word processor make editing BASIC files easier?
2. What purpose is served by using the processor to merge files?
3. What kinds of files cannot be handled by some processors?
4. What is meant by file maintenance?
5. How does a dataset differ from a data base?
6. Is there any information on your disk directory or catalog that tells you whether or not a file is a text or ASCII file?
7. What is an address file, and how is it used by the processor?
8. What is the function of an end-of-address character, and why must each field occupy a separate line?
9. What technique is used when we wish to preserve an existing file that we are reformatting for an alternate purpose?
10. What is the result of giving our input and output files the same name?
11. What procedure is used if we wish to sort merged files?

Assignments:

1. Create an file containing nine or more names and addresses. Use the following format.

Smith, Mary	12 Willow Rd	Front Royal, Virginia	22630
Cole, Howard	3 Red Drive	Penns Grove, New Jersey	08069
Gold, Jack	86-14 Third Ave.	Elnora,	New York 12065

2. Reformat this file in the word processor so that mailing labels can be printed. Use three across. Your new file should look like this.

Ms. Mary Smith	Mr. Howard Cole	Mr. Jack Gold
12 Willow Rd.	3 Red Drive	86-14 Third Ave.
Front Royal	Penns Grove	Elnora
Virginia 22630	New Jersey 08069	New York 12065

3. Create a payroll file of ten or more employees. Input last name and hours worked for each of five days. Give each employee a different hourly rate and calculate gross salary. Load this file into the word processor and add a social security number for each employee.

Be careful to rename the new file so as not to destroy the old one. If your word processor can merge, reformat into an address file and send each employee a letter. Use whatever information you like, but prepare a letter for each employee.

4. Create a file of important dates to remember. Include birthdays, anniversaries, and holidays. Load this into the word processor.

Rearrange dates into monthly sections. Use this file to print a neatly formatted calendar of important dates to remember.

Appendix A1

TYPES OF VARIABLES: REVIEW

Performance Objectives: Students will be able to:

- . explain how a computer stores and labels information
- . define and utilize numeric and string variables
- . describe and use assignment statements with numeric and string variables

Vocabulary: Review: assignment, LET, memory location, numeric variable, string variable

Motivation:

- . What are the values of set A if set A is the set of all whole numbers?
- . Suppose that $A + 7 = 12$. For what values of A will that statement be true?
- . If $A + 7 = 12$, could $A = 9$?
- . If A and B are whole numbers, what are the values of A and B that make the statement $A + B = 12$ true? If we say that $A = 2$ what must be the value of B?

Aim: How are words and numbers represented in a computer?

Development:

1. How could a computer store the information that $A = 2$?
 - . $A = 2$ would be converted into code the computer can use. This information would be stored in a unique memory location inside the computer. Each storage has a unique memory location. The value 2 would be stored in memory location A.
2. What is an assignment?
 - . An assignment is a mapping which is used within a computer to keep track of the location of a particular piece of information.
3. How is an assignment indicated within a computer program? (by the use of the statement LET)
 - . How could we assign the value of 2 to A? ($LET\ A = 2$, or $A = 2$). The variable name is always written first, followed by an equal sign and finally by the value which is to be assigned to that variable.
4. What do we call the A in our sample? (a variable)
 - . A variable is a symbol that represents a number or expression.
 - . It labels a location in memory. The computer uses this device as a means of identifying a value stored in its memory.

5. What type of variable is A if A = 2 ? (a numeric variable)

- . A numeric variable names a memory location for a number or an arithmetic expression. It initializes a value for a particular variable.

6. In BASIC, a numeric variable can be: (1) a single letter of the alphabet; or (2) a letter followed by another letter or digit, such as A, BX, N9, R1, X, C; or (3) two letters of the alphabet.

- . Many computers only recognize the first two characters. (SUM1 and SUM2 would be treated as labels for the same memory location, SU, by those machines.)
- . Let us examine a program that stores numeric values for variables and let us record what the computer retains in its memory as the program is executed.

PROGRAM	COMPUTER'S MEMORY				
	A	B	C	D	E
10 Let A = 12	12				
20 Let B = 16		16			
30 Let C = 0			0		
40 Let A = 6	6				
50 Let D = 1				1	
60 Let E = A					6

(Review the use of line numbers to show the logical progression of the program.)

- . What happened in line 40? (The value of A, which was originally 12, was reassigned to have a value of 6. Stress that the original value was no longer held but was replaced by the current value.)

7. Suppose we wish to store a person's name in a memory location.

- . Would the memory location be labeled with a numeric variable? (No, because a computer labels only numeric values with numeric variables. It cannot use numeric variables to label word values.)

- . What type of variable could then be used? (a string variable)

- . A string variable is a variable used to represent a series of alphanumeric characters.

- . In BASIC, a string variable begins with a single letter or two letters and ends with a \$ (A\$, A2\$, XY\$ are examples of string variables.)

- . Numbers stored as strings cannot be used in calculations.

8. How could we store the message HELLO in memory location M\$? (M\$="HELLO") (The alphanumeric value assigned to the string variable is always enclosed by quotation marks in an assignment statement.)

9. How could we write an assignment which would store our name, address, and phone number in separate memory locations A\$, B\$, and C\$?

```
LET A$ = "BEN FRANKLIN"  
LET B$ = "234 Liberty Square"  
LET C$ = "123-1776"
```

Applications:

1. Compare and contrast string variables.
2. Explain why the following two statements will yield identical results:

```
Let A = 7           A = 7
```

Summary:

1. How does a computer store data and programs?
2. What are the different types of variables?
3. What are some of the characteristics of string and numeric variables?
4. What is an assignment statement?
5. What are the forms and uses of assignments?

Assignments:

1. Determine whether the following are valid assignments:
 - (a) A\$ = "FRED"
 - (b) X = 14
 - (c) M = Ann
 - (d) F\$ = 1234
 - (e) Z\$ = "234 Lightner Drive"
 - (f) R = R - 5
 - (g) E\$= 727-5000
2. Write the assignment statements that could store your name, date of birth, and age in memory locations N\$, B\$, A\$.
3. Write assignment statements that will store the set of even numbers, beginning with 0 and ending with 10.

Appendix A2

THE PRINT STATEMENT: REVIEW

Performance Objectives: Students will be able to:

- . identify the function of the PRINT statement
- . use the PRINT statement for display of a message
- . use the PRINT statement to evaluate an arithmetic expression
- . identify the order in which arithmetic expressions are evaluated

Vocabulary: Review: CLS, colon, comma, immediate mode, LIST, numeric variable, operators (+, -, *, /), PRINT, programming mode, RUN, string variable, syntax error

Motivation:

- . If A\$ = "HAPPY BIRTHDAY", what instructions can we give the computer so that it will display this message on its monitor?
- . How can we give the computer explicit instructions to output a message?
- . How can we give the computer instructions to perform a mathematical calculation and then output the result?
- . How will this mathematical calculation be carried out?

Aim: How can we use the PRINT statement to display string as well as numeric messages in the programming mode as well as in the immediate mode?

Development:

1. We know that there are two types of variables:

Numeric variables (those variables that represent a number or an arithmetic expression)

String variables, or alphanumeric variables (those variables that represent a series of alphanumeric characters)

- . If we wish to display the value of the variable on the monitor, what command must we give the computer? (PRINT)

2. X\$ = "MAY YOUR DAY BE BRIGHT"

- . What command must be given to the computer so this message will be displayed on the monitor? (PRINT X\$ or PRINT "MAY YOUR DAY BE BRIGHT")
- . What instructions can we give the computer so that it will display the message "GONE FISHING"? (PRINT "GONE FISHING")
- . The PRINT statement may be used to display a string on the monitor by using the command PRINT followed by quotation marks around the string

(for example, PRINT "HELLO") or by labeling the string with a string variable and then using the command PRINT followed by the string variable (for example, H\$ = "HELLO":PRINT H\$)

3. Suppose the variable is a numeric variable. What command could be used to display the value of the variable on the monitor? (the PRINT statement)

- . How could we display the number 123 on the monitor? (PRINT 123)
- . If X = 75, what programming statement could be used to display the value of X on the monitor? (PRINT X)
- . The PRINT statement may be used to display a number on the monitor by using the command PRINT followed by the number (for example, PRINT 6) or by labeling the number with a numeric variable and then using the command PRINT followed by the numeric variable (for example, T = 2 : PRINT T)..

4. What programming statements could be used to display your name on the monitor, followed by your age on the next line?

Answer:

```
PRINT "ABE LINCOLN"  
PRINT 48
```

- . In what two different ways can these instructions be given? (in the programming mode and in the immediate mode) The programming mode allows us to give the computer a series of instructions that it keeps in its memory for later performance at our command.
- . The immediate mode lets the user give the computer a direct command and see the results immediately.
- . How are the lines written in the programming mode?

Answer:

```
10 PRINT "ABE LINCOLN"  
20 PRINT 48
```

- . How are the lines written in the immediate mode?

Answer:

```
PRINT "ABE LINCOLN"  
PRINT 48
```

- . Stress that line numbers must accompany a PRINT statement in the programming mode.
5. How can we use the immediate mode to solve the problem $8 + 6$?

Answer:

PRINT 8 + 6

- . Show that by using the command PRINT followed by the mathematical expression 8 + 6 we will find the answer on our screen after pressing the CARRIAGE RETURN key.

- . Review the operators used to perform mathematical operations:

addition: +
subtraction: -,
multiplication: *
division: /

- . What would the outcome have been if the following statement had been typed into the computer?

PRINT " 8 + 6 "

(8 + 6)

- . Why wasn't the answer 14 printed out? (If words or other symbols are enclosed in quotes, the computer sees them as strings and prints them exactly as they were entered.)

- . Type the following program into the computer and study the results:

10 A\$ = "SUN"
20 B\$ = "LIGHT"
30 Print A\$ + B\$
40 END

- . What was the output? (SUNLIGHT)

- . When two strings are "added" they are joined together as a single word.

6. What would the outcome be if the following statement were entered?

PRINT HELLO

Answer:

(0)

- . Stress the fact that the computer treated the word HELLO as the numeric variable, HE, because no quotation marks were placed around the word. As a result the variable HE had a value of 0 because no other value was initially stated for it.

If the value of a numeric variable is not initialized, the computer sets the initial value at 0.

7. When we do the following problem, in what order are the arithmetic operations performed?

$$8 + 4 \times (6 - 8) / 4 [3$$

Answer:

Parenthesis

Exponents

Multiplication or division (in the order they appear)

Addition or subtraction (in the order they appear)

- . Write the following mathematical statement in the immediate mode.

$$9 + 8 - (4 + 6 \times 3^2) / 5$$

Answer:

$$\text{PRINT } 9 + 8 - (4 + 6 * 3 [2] / 5$$

- . Evaluate each expression using paper and pencil. Then, use the computer in the immediate mode to check your results:

$$35 + 8$$

$$6 \times 9 \times 4$$

$$8 \times 10 - 5 \times 4$$

$$3 / 4 - 6$$

$$30 - 6 [2 / 4$$

- . Write each of the following algebraic statements as equivalent BASIC statements

$$P = 2 (L + W)$$

$$A = \frac{1}{2}BH$$

$$X = 4B + 3M - 7D$$

$$C = D(2R - B)^2$$

8. Now that we have performed these problems, the screen is very cluttered. The information on the top of the screen has been pushed off.

- . The upward movement of the contents of the screen is called scrolling.

- . How can we clear the screen so we can try some more examples?

. (Review the clear screen function: IBM - CTRL/HOME; PET - (SHIFT)CLR/HOME; TRS-80 - CLEAR/CLS; Apple - HOME.)

Applications:

1. A movie theater charges admission of \$4.50 for adults and \$2.75 for children. How much revenue will the theater earn if 250 adults and 180 children attend the performance?
2. Write a computer program that will solve this problem, using sentences to define the output.

Summary:

1. Compare and contrast the programming and the immediate mode of the computer.
2. Compare and contrast the way a string variable and a numeric variable must be presented within a program so that they can be displayed on the monitor.
3. Review the order in which operations are performed by the computer.

Assignments:

1. If $A = 8$, $B = 23$, and $C = 5$, write a program that will display the results of the following arithmetic statement:

$$Y = 3A + 4B - 8(C + B)^2$$

$$Z = \frac{A + 4}{3B - C}$$

2. Write a program that will print the word "GOOD", clear the screen, print the word "BYE", clear the screen, print the word "GOODBYE".

3. Name the operation that will be performed first in each of the following statements:

$$\begin{aligned} &18 - 3 * 4 \\ &16 / 9 - 5 + (12 + 3) \\ &(2 - 1) * 3 / 5 \\ &2(9 * (4 + 3 * 2) - 8) \\ &6 * 5 / 8 [3 + 4 \end{aligned}$$

4. The perimeter of a rectangle can be expressed by the formula $P = 2 \times L + 2 \times W$ where L is the length of the rectangle and W is the width of the rectangle. Write a program that will compute the perimeter if the length is 7 meters and the width is 6 meters.

Appendix A3

FURTHER USES OF THE PRINT STATEMENT: REVIEW

Performance Objectives: Students will be able to:

- . describe the output resulting from the use of commas in a PRINT statement
- . describe the output resulting from the use of a semicolon in a PRINT statement
- . describe the vertical and horizontal screen format
- . use the PRINT TAB command
- . use the PRINT statement as a means of creating "blank" lines within the output of a program

Vocabulary: New: PRINT TAB, horizontal spacing, vertical spacing
Review: CLS, CLEAR, LIST, PRINT, print format, RUN

Motivation:

- . Forecast the display on the monitor of the following programming statements:

```
PRINT " H E L L O "
```

```
PRINT" HELLO"
```

```
PRINT  "HELLO"
```

```
PRINT"      HELLO"
```

- . What was the effect of the space within the quotes?
- . What was the effect of the space outside the quotes?
- . How might we print the same message centered on a line?
- . How might we print the same message centered on the screen?

Aim: How may we format print messages to be displayed at specific locations on the monitor?

Development:

1. Examine the following programs. Forecast the outcome of each program.

PROGRAM A

```
10 CLS
20 X = 8
30 Y = 9
40 Z = X + Y
50 PRINT X
60 PRINT Y
70 PRINT Z
80 END
```

PROGRAM B

```
10 CLS
20 X = 8
30 Y = 9
40 Z = X + Y
50 PRINT X Y Z
60 END
```

Answer:

outcome is:
8
9
17

outcome is:
0

- Why was zero the outcome in program B?

Answer:

The statement PRINT X Y Z is treated as the statement PRINT XYZ. Since the variable XYZ is not initialized the computer gives it an initial value of 0. In program A the computer printed the values for X, Y, and Z. These values were initialized within the program when $X = 8$, $Y = 9$ and $Z = X + Y$.

2. How might we have printed line 50 in program B so that X, Y, and Z would have been displayed on the same line?

- Let's rewrite line 50 in the following way. Examine the outcome when this program is RUN.

```
50 PRINT X, Y, Z
```

- What was displayed when this program was run?

Answer:

8 9 17

- Introduce the idea of print zones. When the variables in a PRINT statement are separated by commas, the comma will make the computer proceed to a new zone before printing. The columns are print zones. These zones are predetermined by the computer. They depend on the number of characters that can be typed across a screen and differ from computer to computer. If the value of the variable is greater than the number of characters within a zone, the value will be printed to its end and the value of the next variable will be printed in the next available zone.

- How many zones does your computer utilize for printing?
- Type in the following command, which will illustrate where the zones are located within your computer:

```
PRINT A,B,C,D,E,F,G
```

Now, go back to program A. Rewrite lines 50 to 70 as follows:

```
50 PRINT X,
60 PRINT Y,
70 PRINT Z
```

What would the screen look like?

Answer:

```
8           9           17
```

3. How might we have rewritten the program so that the output displayed on the monitor might have appeared as follows:

```
8 9 17
```

- . If we had used a semicolon instead of a comma, the values of the variables would have been printed next to each other on the same line. Positive numbers will always be printed with a space in front of them. The space stands for the sign, which conventionally is not printed for positive numbers.

How can program B be rewritten so that we could get the above output?

Answer:

```
10 CLS
20 X = 8
30 Y = 9
40 Z = X + Y
50 PRINT X;Y;Z
60 END
```

4. Now review the comma and semicolon using string variables.

- . Forecast the output of the following two programs:

PROGRAM C

```
10 CLS
20 A$ = "SUN"
30 B$ = "DAY"
40 PRINT A$, B$
50 END
```

PROGRAM D

```
10 CLS
20 A$ = "SUN"
30 B$ = "DAY"
40 PRINT A$; B$
50 END
```

Answer:

```
SUN           DAY           SUNDAY
```

- . Do the rules of commas and semicolons hold for string variables as well as numeric variables? (yes)
- . What is the difference in screen format between a semicolon and a comma in a program? (A comma will print the message in zones while a semicolon will print the messages in the next available space.)

5. Suppose we wanted to center our message instead of print it in the next available space or next available zone. How might we accomplish that?
 - . Show that the center of a horizontal line would have to be located along with the center character of the message to be printed.
 - . Print the message READY in the center of the line. (If the screen is 64 characters long, the center position is at 32. In the word READY, the A is the center letter. Thus, the A in the word READY must be in the 32nd space on the line and, as a result, the R is in the 30th space on the line.)
 - . We can then give the following command:


```
PRINT"                      READY"
```

(Press the space bar 29 times; then type the word READY.)
 - . Write the command that will center the message HAPPY NEW YEAR on the line


```
PRINT"                      HAPPY NEW YEAR"
```

(Press the space bar 26 times; then type the message.)
6. What easier way is there to advance to the 27th space on the screen, other than pressing the space bar 26 times?
 - . Introduce the PRINT TAB statement as a function similiar to the tab function on an ordinary typewriter.
 - . To print HAPPY NEW YEAR in the 27th space we could give the computer the command


```
10 PRINT TAB(27) "HAPPY NEW YEAR"
```
 - . How could you print your name in the center of the line using the PRINT TAB function?
7. Suppose you want to type your first name in the center of the screen. How could we accomplish that?
 - . We could use the PRINT TAB function to center the name on a line, but how could we locate the center line? (We could use the PRINT statement alone to create a blank line and then use the PRINT TAB function.)
 - . Each computer has a specific number of characters horizontally as well as a specific number of lines vertically.
 - . Distribute copies of the video work sheet for the specific computer used and explain how specific locations can be found and what a location's address is.
 - . Write the command that will print the message "GOODBYE" in the extreme lower left corner of the screen.

Applications:

1. Mrs. King wishes to set up a computer program for listing her students' grades in columns. Set up the PRINT format with the following headings: NAME, TEST 1, TEST 2, TEST 3, AVERAGE.
3. Allow 30 spaces for the students' names and assume that the highest score per test would be 100. Also, allow a margin of 5 spaces before the name is listed.

Summary:

1. How does the use of a comma compare with that of a semicolon within a PRINT statement?
2. How can you locate a specific position on a line?
3. How can you locate a specific position on the screen?

Assignments:

1. Write a program that lists the name, address, and phone number of five friends, including the following headings:

NAME	ADDRESS	PHONE
------	---------	-------
2. Write a program that will print out the first 10 even numbers in the zone format.
3. Write a program that will print out your name in the middle of the screen, remove it, and then print a message of your choice in the middle of the screen.
4. The formula to determine the volume of a rectangular prism is $V = l \times w \times d$, where l is length, w is width, and d is depth of the prism. Write a program that will determine the volume of a rectangular prism whose length is 1.4 cm., width is 2.6 cm., and depth is 5 cm. The output should resemble the following:

LENGTH	WIDTH	DEPTH	VOLUME
1.4 cm.	2.6 cm.	5 cm.	-- cu.cm.

Appendix A4

INPUT STATEMENTS: REVIEW

Performance Objectives: Students will be able to:

- . create an interactive program
- . use an INPUT statement with numeric variables
- . use an INPUT statement with string variables
- . recognize the implications of a REDO message
- . use the enhanced INPUT statement (sometimes known as INPUT with prompt statement) as an alternative to PRINT...INPUT statements

Vocabulary: New: enhanced input, interactive program, prompts, REDO
Review: alphanumeric, INPUT, numeric variables, PRINT, use of semicolon in PRINT, string variable

Motivation:

- . If you were asked "What is your name?" what would be a logical response?
- . If the computer asked you for your name, what would you do?
- . Examine the following program. When you run this program will you have an opportunity to respond to the questions the computer asks?

```
10 PRINT "What is your name?"
20 PRINT "What is your address?"
30 PRINT "What is your phone number?"
40 END
```

- . What is needed within this program so that you will have an opportunity to answer the questions the computer asks?

Aim: What are the ingredients of an interactive program?

Development:

1. We would like to develop "reactions" to questions in a computer program.

- . What is needed? (an opportunity to answer the questions) At what point in the program should an answer logically be inserted? (after a question is asked)

2. Let's re-examine our program.

```
10 PRINT "What is your name?"
20 PRINT "What is your address?"
30 PRINT "What is your phone number?"
40 END
```

- . Insert an X wherever you feel you would like to answer the question the computer asks:

```

10 PRINT "What is your name?"
      X
20 PRINT "What is your address?"
      X
30 PRINT "What is your phone number?"
      X
40 END

```

3. We want to create an interactive program, where a user is permitted to interact with the computer in a two-way type of conversation. In our program, we want the message on line 10 to appear, then we want the user to be able to respond to that message by answering the question the computer presented.
 - . In an interactive program the user gives information to the computer and receives information from the computer while the program is being executed. Since the computer is able to receive information externally, data need not be internally supplied within the program.
 - . An interactive program is a program that allows the user to interact with a computer by entering data rather than using preprogrammed values.
4. Let us now create an interactive program which will permit the user to answer the questions the computer asks. These messages the computer asks of the user are known as prompts.
 - . Let us return to our original program. The computer would like to know your name, address, and phone number. Once the computer has this information it will print out the message: Hi (your name). You live at (your address). Your phone number is (your phone number)."

```

10 PRINT "What is your name?"

```

 - . At this point we would like to allow computer input for your answer to the question.
 - . What type of variable will your response need for storage? (string variable)
 - . In order for the computer to accept the response, we must tell it that information will be supplied by the user during the execution of the program. The necessary command is INPUT.
 - . The INPUT statement allows us to assign a value to a variable as we use a computer program.
 - . The INPUT statement places a question mark on the screen and causes execution of the program to pause until the user has entered a response. Once the response is INPUT, the program continues execution.
 - . Now, what does our program look like at this point?

```

10 PRINT "What is your name"
15 INPUT N$
20 PRINT "What is your address"
25 INPUT A$
30 PRINT "What is your phone number"
35 INPUT P$
40 PRINT "Hi";N$;". You live at ";A$; ". Your phone number is ";P$
50 END

```

5. Examine the following program. What would appear on the screen as this program is executed?

```

10 PRINT "What is your last name"
20 INPUT L$
30 PRINT "What is your first name"
40 INPUT F$
50 PRINT "Welcome to computers ";F$;" ";L$
60 END

```

What is your last name
? (student gives response)

What is your first name
? (student gives response)

Welcome to computers (first name last name)

- . Stress the fact that the ? is used by a computer to indicate that it is waiting for a user response to an INPUT statement. It will remain on the screen until a response is supplied.

6. How might a computer ask for your age?

```

10 PRINT "What is your age"
20 INPUT A

```

- . Why was line 20 INPUT A? (A is a numeric variable. Since the question asked requires a response of a number, a numeric variable should be used to name that number.)
 - . Run the program above and type your age in words. What occurs? (A REDO message appears on the screen). Why does this message appear? What is its significance? (It indicates that the value you input for the variable is not the type which the variable was programmed to accept.) In this case, you INPUT a string or alphanumeric variable, while the computer had been told that a numeric variable would be INPUT.
7. Write a program that will ask for your favorite number, your age, and the year you were born. Have the computer then proceed to output your secret number, which will be arrived at by the following formula:

SECRET NUMBER = Favorite number + Age + year of birth

```

10 PRINT "What is your favorite number"
20 INPUT F
30 PRINT "How old are you"
40 INPUT A
50 PRINT "In What year were you born"
60 INPUT Y
70 SN = F + A + Y
80 PRINT "Your secret number is";SN
90 END

```

8. To consolidate the program, we may combine the PRINT and INPUT statements into a single programming line.

- Examine lines 10 and 20 above, then compare them to the following single line:

```
10 INPUT "What is your favorite number"; F
```

- What occurred in this new line? (A single INPUT, followed by the message to be printed, then followed by the name of the variable to be input was used instead of the PRINT statement followed by an INPUT statement.)
- This is an example of an enhanced INPUT statement.
- An enhanced INPUT statement combines a PRINT and INPUT statement into a single command. The BASIC word INPUT is followed by the message to be printed and enclosed within quotation marks. This is followed by the name of the assigned variable and represents the response the program user gives. (Don't forget the semicolon.)
- Rewrite the following program using the enhanced input statement instead of the PRINT and INPUT statements.

PROGRAM A

```

10 PRINT "What is the first number to be added?"
20 INPUT A
30 PRINT "What is the second number to be added?"
40 INPUT B
50 PRINT "What is the third number to be added?"
60 INPUT C
70 PRINT "The sum of ";A;" + ";B;" + ";C;" is"; A+B+C
80 END

```

Acceptable answer:

```

10 INPUT "What is the first number to be added";A
20 INPUT "What is the second number to be added";B
30 INPUT "What is the third number to be added";C
40 PRINT "The sum of ";A;" + ";B;" + ";C;" is "; A+B+C

```

9. We could have simplified this program even further. How? (Suggest that

we might have written a single question in a form similar to "Type in three numbers to be added".)

- . Show that a single INPUT statement may be used to insert values for more than one variable at a time.

- . Replace lines 10 to 30 with the following single line:

```
INPUT "Type in three numbers to be added";A,B,C
```

- . Review the fact that the ? is used to indicate the computer is awaiting a response from the user. In this case, the computer is waiting for three numbers to be input, either one at a time or all three at once. If the numbers are input one at a time, the computer will show a ? for the first number to be input, and when input, will indicate a new ? or set of question marks for all other variables to be input from that single line. (On some brands of computer, only a response including three numbers will be accepted.)

Applications:

1. Write a program to determine the area of any circle, knowing that the formula for the area of a circle is

Area = Radius x Radius x 3.14

- . The prompts should ask for the radius and the final message should state, "The area of the circle is"---

Acceptable answer:

```
10 INPUT "What is the radius of the circle";R
20 A = R [ 2 * 3.14
30 PRINT "The area of the circle is ";A
```

Summary:

1. What do we call a program that permits the user to interact with the computer while the program is being executed?
2. What useful programming tool aids the user in giving a correct response to an INPUT statement?
3. Compare an INPUT statement and an enhanced INPUT statement.

Assignments:

1. Write a program that will determine a student's average of five test scores and require the user to INPUT the scores all at once. Make the computer print out the five scores and the average, in columns.
2. Write a program that will compute a baseball player's batting average, once the user INPUTS the number of times the player was at bat and the number of times a hit was scored. The output should include the player's name and batting average.

Acceptable answer:

Batting average = hits scored / times at bat

3. Write a program that will calculate the age differences among three or more people. Include prompts that will cause the user to enter the ages in a specific order, youngest to oldest or oldest to youngest. Then have the machine compute the differences in age among the people and print out those differences.

Appendix A5

FLOWCHARTING: REVIEW

Performance Objectives: Students will be able to:

- . identify the meaning of the flowchart symbols: oval, rectangle, parallelogram, diamond
- . write a flowchart to solve a given verbal problem
- . interpret and trace a flowchart involving decision statements as well as conditional statements

Vocabulary: Review: oval, rectangle, parallelogram, input, output, decision box, conditional branching, connector

Motivation:

- . Put the following directions in the correct order:

- (a) Put key in ignition
- (b) Open door
- (c) Walk to car
- (d) Go to the driver's side
- (e) Turn key in ignition
- (f) Sit in the driver's seat

- . Why do we follow this sequence?
- . What problem does this sequence solve?
- . How do we solve it?

Answer:

Identify the problem
Analyze each step of the problem (stepwise refinement)
Break down the problem into steps

- . How does this process apply to writing a program?

Aim: How can a flowchart aid in creating a program?

Development:

1. To be a member of the Honors Squad you must have an average of at least 88% in your major subjects (English, Mathematics, Social Studies, and Science). What steps would you use to determine if you are eligible to become a member of the Honors Squad?

Answer:

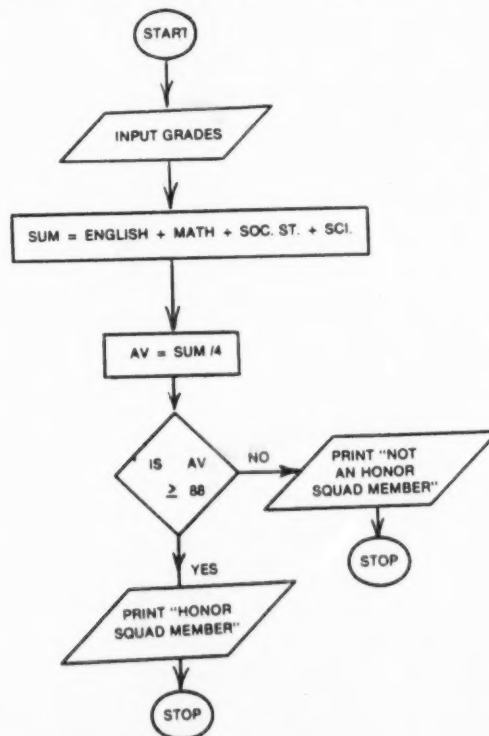
obtain your four grades
add the grades

divide the sum by four
state the average
compare the average to 88

2. Let's look at the flowchart symbols that can be used to create a flowchart for this problem:

- . What symbol is used to begin or end a flowchart? (oval)
- . What shape is used to indicate information that must be input? (parallelogram)
- . What shape is used to indicate the processing of information? (rectangle)
- . What shape is used to indicate the need for a decision? (a diamond)
- . What shape is used to indicate the output of a solution? (parallelogram)
- . What symbol is used to show the flow from one step to the next? (an arrow)

3. Let us now draw a flowchart that will illustrate these steps:



4. Let's review how the symbols were used:

Terminals - oval (indicates the beginning or the end of a computer program)

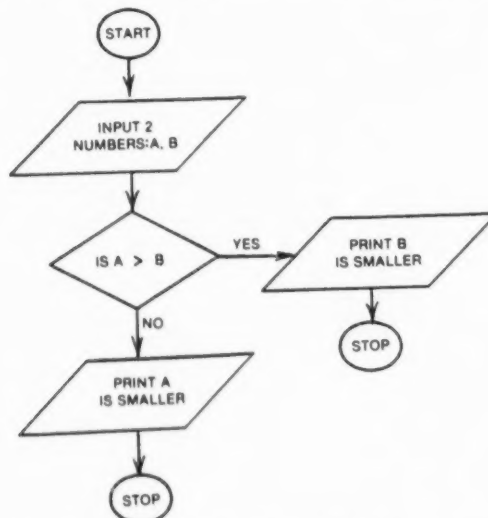
Input/Output (I/O) - parallelogram (indicates information is coming in or going out)

Processing - rectangle (indicates computer is acting upon the information supplied)

Decision - diamond (if the condition in the box is true, follow the "Yes" arrow; if false, follow the "No" arrow.)

Lines of flow - arrow (indicates sequence of steps)

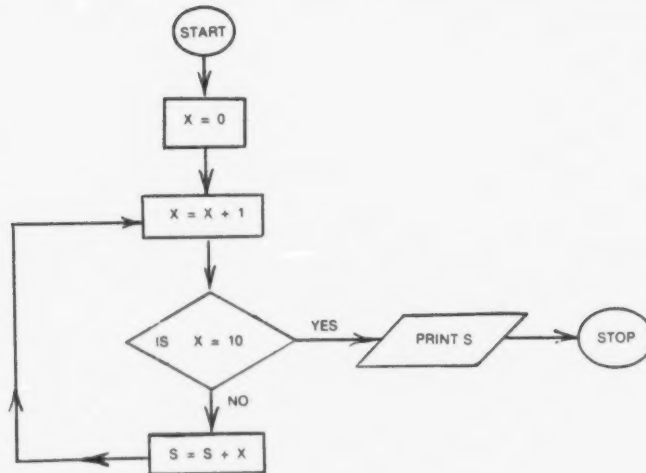
5. Why are there two branches from the diamond? (A decision statement offers two pathways from the decision, based on whether the condition in the box is true or false.)
6. What is the purpose of a flowchart? We use flowcharts to solve a problem by arranging the necessary steps in a logical sequence.
 - . Flowcharting symbols have been standardized to reflect the major components of any computer system: INPUT, PROCESSING, OUTPUT.
 - . Using the flowchart, the processing may be translated into a computer programming language.
7. Write a flowchart to PRINT the smaller of two unequal numbers.



8. Examine the flowchart at the top of the next page and determine the output.

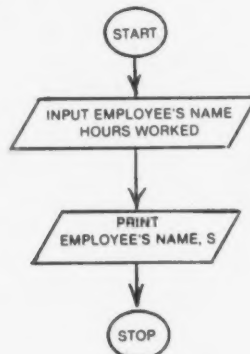
(The flowchart illustrates the addition of the first nine positive whole numbers, whose sum is 45.)

- . What is the purpose of the arrow from $S = S + X$ to $X = X + 1$ (It forms a loop.)
- . How many times will this loop be executed? (9 times)



Applications:

1. Write a flowchart that will INPUT an employee's name and the number of hours worked and will then compute the employee's salary for the week at a rate of pay of \$4.75 per hour.



2. Rewrite this problem using a flowchart.

```

1  L = 12
2  W = 17
3  P = 2 (L + W)
4  Print P
  
```

Summary:

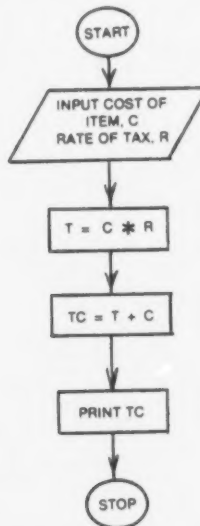
1. What is the purpose of a flowchart?
2. What symbols are used in a flowchart?
3. How can a flowchart be helpful to a programmer?
4. What do the flowcharting symbols represent in computer terminology?

Assignments:

1. Flowchart the following activities:

Mailing a letter
Making a cup of tea
Finding the circumference of a circle

2. Examine the following flowchart and identify the problem it is designed to solve.



3. Write a flowchart that will compute a person's weekly salary if that person earns \$5.25 per hour for the first 35 hours of work per week and earns \$8.30 per hour for any hours worked beyond the 35 hours.
4. Write a flowchart that will INPUT a baseball player's times at bat and the number of hits and then will OUTPUT the player's batting average. (batting average = number of hits / times at bat)

Appendix A6

READ/DATA STATEMENTS (PART I): REVIEW

Performance Objectives: Students will be able to:

- . provide data with READ/DATA statements
- . distinguish between INPUT and READ/DATA statements
- . identify common errors made in using a READ/DATA statement
- . understand the significance of an out of data error

Vocabulary: New: out of data error

Review: DATA, GOTO, INPUT, READ, variables

Motivation:

- . Write a program that asks for a person's name and the number of hours worked per week. Use INPUT statements to obtain the information needed. Output the person's name and the salary earned if the rate of pay was \$4.63 per hour.
- . If there were five employees how many times must the program be RUN?
- . If the only information that varied from week to week was the number of hours worked, what variables could remain the same?
- . Is there a way these variables could be saved within the program itself so that the employees names and hourly rates of pay would not have to be requested from the user each time the program was RUN?
- . Could this data be stored within a program?

Aim: How can data be stored within a program so that it can be readily accessed for different purposes during program execution?

Development:

1. Let us review the original program that was written

```
12 INPUT"EMPLOYEE'S NAME ";N$
20 INPUT"Number of hours worked ";H
35 S = h * 4.63
40 PRINT "EMPLOYEE'S NAME","SALARY"
47 PRINT N$,S
60 END
```

We know that to execute this program for five employees, we would have to RUN the program five times each week.

We would retype the employees' names each time the program was RUN even though those names (N\$) did not change each time.

Is there a way to eliminate this task, so that the employees' names could be included within the program and would not have to be retyped each time the program is RUN? (We can store the data within a DATA statement.)

2. Let us modify the program so that the names of the five employees could be inserted into the program itself and would not have to be INPUT each time the program is executed.

```
10 DATA Jones, Edwards, La-Touche, Reilly, Torm
20 INPUT "Number of hours worked ";H
30 S = H * 4.63
40 PRINT "EMPLOYEE'S NAME ", "SALARY"
50 PRINT N$,S
60 END
```

Now that the data is stored within a DATA statement in the program itself, how can this information be accessed during execution of the program? (by using a READ statement)

Note: In some versions of BASIC, quotation marks are necessary around each DATA element (for example, "MARY", "BOB")

Where shall we include the READ statement within this program so that the computer will READ the employee's name first then ask for the hours that employee worked? (between lines 10 and 20)

3. Let us examine our program now.

```
10 DATA Jones, Edwards, La-Touche, Reilly, Torm
15 READ N$
20 PRINT "Number of hours ";N$;" worked"
25 INPUT H
30 S = H * 4.63
40 PRINT "EMPLOYEE'S NAME", "SALARY"
50 PRINT N$,S
60 GOTO 15
70 END
```

Explain that READ N\$ tells the computer to look for a DATA line. The computer then picks out the first value in the first numbered DATA line. The machine then continues until it reaches line 60. What happens at line 60?

Explain that the GOTO sends the program back to line 15 and again continues the execution of the program. The computer then READS the next value in the DATA line as the value for N\$. The process continues until all values in the DATA line have been utilized.

A DATA statement may appear anywhere within the program and need not appear before the READ statement that needs its data. When a computer encounters a READ statement, it searches for a DATA statement and READS the next available information.

4. Run this program and record the results if the employees worked the following number of hours, respectively: 32, 16, 29.5, 4, 40.

How many times will the computer go through the process? (5 times) What will happen on the sixth try? (No more data will be available.)

How does the computer indicate this fact? (an out of data error message appears on the screen.)

5. Write a program that will write the following message to five friends named Ed, Amy, Jo, Baruch, and Juan:

Dear -----

Thank you for your gift.

Love,
Fran

Acceptable answer:

```
10 READ A$
20 PRINT "DEAR ";A$
30 PRINT
40 PRINT "      Thank you for your gift"
50 PRINT
60 PRINT "                      Love,"
70 PRINT "                      Fran"
80 GOTO 10
90 DATA Ed, Amy, Jo, Baruch, Juan
100 END
```

Review the fact that a DATA statement may appear anywhere in a program and that its location in no way affects the outcome of the program.

6. Type the following program into a computer and record the results:

```
10 READ A
20 PRINT "List the four test scores for ";A
30 INPUT B,C,D,E
40 AV = (B + C + D + E)/4
50 PRINT " the average grade for ";A;" is ";AV
80 GOTO 10
90 DATA Smith, Able, Sanchez, Lee
100 END
```

Why won't this program RUN? (The READ statement is incorrect because the DATA to be READ is string DATA while the command asks the computer to READ it into a numeric variable.)

How may this be corrected? (10 READ A\$)

Stress that the variable type employed in a READ statement must correspond to the information supplied within its corresponding DATA statement.

7. What would be the output of the following program?

```

10 READ A
20 READ B
30 C = A * B
40 PRINT A,B,C
50 GOTO 10
60 DATA 2,5,3,2,6,7
70 END

```

Answer:

2	5	10
3	2	6
6	7	42

After the computer executed line 10, where did it search for DATA so that it could execute line 20? (in the next available DATA entry of line 60)

Why did this occur? (It occurred because when a computer detects a READ statement in its program stream, it searches for the next available piece of data within a DATA statement and then labels that data according to the variable in the READ statement. As a result, the first number in the DATA statement was labeled A, the second was labeled B, the third A, and so forth.)

Applications:

- Using READ/DATA statements, write a program that will list the amount of interest earned in two years on a \$2000 savings bond if the rate of interest per year is:

- (a) 0.075
- (b) 0.09
- (c) 0.12
- (d) 0.068
- (e) 0.055

Remember, $\text{interest} = \text{principal} \times \text{rate} \times \text{time}$

- Print the rate of interest and the interest earned, as output.

Summary:

- What are READ/DATA statements?
- When a READ statement appears within a program, must a DATA statement appear also?
- What is the relationship between the DATA to be READ into a specific variable and the type of DATA presented in the DATA statement?
- What must separate data values within a DATA statement?
- What problem is indicated by an out of data error?

Assignments:

1. Determine the output that would result from RUNNING the following program.

```
10 READ A$
20 READ B
30 READ C
40 READ D
50 AV =( B + C + D)/3
60 PRINT "Name","Average"
70 PRINT A$,AV
80 GOTO 10
90 DATA Jones, 80, 92, 68, Lee, 60,100, 100
110 DATA Marks, 84, 79, 88
120 END
```

2. Write a program using READ/DATA statements that will store the names of five hardware items within a data statement. Then have the machine ask the user for the current price of each item in three different stores. The computer should compute the average price of each item and then print out the name of the item and the average price.
3. Create a program that will store the name, address, and phone numbers of 10 friends and then print out a party invitation to each of them.

Appendix A7

READ/DATA STATEMENTS (PART II): REVIEW

Performance Objectives: Students will be able to:

- . provide values for more than one variable using a READ/DATA statement
- . write a program using READ/DATA statements involving more than one variable
- . write a program using DATA statements that accomplishes more than one function
- . use a REM statement to document a program

Vocabulary: New: RESTORE
Review: GOTO, READ/DATA, REM

Motivation:

- . Examine the following program

```
10 READ A$
20 PRINT "How many hours were worked by "; A$
30 INPUT H
40 INPUT "Wage per hour "; W
50 S = H * W
60 PRINT A$; " will earn $"; S; " for "; H; " hours worked"
70 GOTO 10
80 DATA ED LEWIS, MARY CUEVAS, LEW AYRES, JANE SMYTH
```

- . What is the purpose of the READ statement in line 10? (to avoid the need for a user INPUT of employee names)
- . Why is this important? (It is faster, more efficient)
- . What other variables might remain constant in this program (hourly wage, number of hours worked)
- . Suppose an employee's hourly wage remained the same. Would it be possible to include this information within the same DATA statement? (yes)
- . How could this be accomplished?

Aim: How can we use READ/DATA statements to store DATA for more than one variable?

Development:

1. Where might we include the additional data within the program (within the existing DATA statement)
2. Suppose each employee's hourly rate of pay was as follows:

EMPLOYEE'S NAME	HOURLY RATE OF PAY
ED LEWIS	\$ 9.25
MARY CUEVAS	\$11.00
LEW AYRES	\$ 6.63
JANE SMYTH	\$ 6.98

How might this information be included within the program? Is it possible to include the hourly rate of pay within the same DATA statement?

Review the fact that each item of information within the DATA statement must be followed by a comma except at the end of a DATA line.

Review the fact that all information within a DATA statement must be compatible in type with each READ variable that is reached by a program. All numeric variables in a READ statement must sequentially correspond to numeric values in a DATA statement, and all string variables in the READ statement must correspond to string values within the DATA statement.

The computer's internal pointer advances on each pass, thus guaranteeing that the correct value is assigned to the variable each time it is needed. As a result, the READ variable takes on the next available value from the DATA statement. If the data in a DATA statement is exhausted the data pointer moves to the next available DATA statement and continues in this fashion until it either runs out of data or the program terminates normally.

- Let us now insert an hourly rate of pay within the already existing DATA statement in our sample program. Where might the rate of pay be stored? Should it follow all of the employees' names? Should it appear before the names or perhaps somewhere else within the DATA statement? (Each rate of pay should appear either immediately before or after each corresponding employee's name.)

How can our DATA statement be revised?

Acceptable answer:

80 DATA ED LEWIS, 9.25, MARY CUEVAS, 11, LEW AYRES, 6.63, JANE SMYTH, 6.98

- Run the program now, using line 80 as we have revised it. What happens? (The computer will ask for the number of hours worked and the wage per hour for each employee in turn. It then will then compute the salary and print out each employee's name and salary.)

What must be done to make use of the data now included within the DATA statement that contains the names and the hourly rates of pay for each employee? (Line 40 must be eliminated and in its place a READ statement must be written.)

Rewrite the program to correct this problem.

Sample program:

```
10 READ A$
20 PRINT " hours worked by ";A$
30 INPUT H
40 READ W
50 S = H*W
60 PRINT A$;"will earn $";s;" for ";H;" hours worked"
70 GOTO 10
80 DATA ED LEWIS, 9.25, MARY CUEVAS, 11, LEW AYRES, 6.63, JANE SMYTH,
6.98
```

5. Create a program that will READ the supplied data, add the numbers in pairs and then PRINT out the original numbers as well as their sum. Use READ/DATA statements to INPUT the numbers.

DATA 5, 16, 9, 27, 12, 1

Acceptable answer:

```
10 READ A
20 READ B
30 PRINT A,B,A+B
40 GOTO 10
50 DATA 5,16,9,27,12,1
60 END
```

6. Now, let us examine a similar program. Determine the output which would result from RUNNING this program:

```
10 READ A,B,C
20 PRINT "A=";A;" B=";B;" C=";C
30 RESTORE
40 READ D,E,F
50 PRINT "D=";D;" E=";E
60 PRINT "F=";F
70 DATA 2,4,6
80 END
```

Answer:

A=2 B=4 C=6
D=2 E=4
F=6

How could the computer READ six values when only three were supplied by the DATA statement? (The command RESTORE sends the internal pointer back to the beginning of the data in the DATA statement.)

Stress the fact that by using the RESTORE command data need be listed only once for use at different times and may then be reused whenever the RESTORE command is issued.

7. Now let us revise our program again. Edit the program so that it will appear as follows.

```

10 READ A,B,C
20 PRINT "A=";A;" B=";B;" C=";C
25 REM THE DATA WILL NOW BE REUSED
30 RESTORE
40 READ D,E,F
50 PRINT "D=";D;" E=";E
60 PRINT "F=";F
70 DATA 2,4,6
80 END

```

What was the outcome? (the same outcome as before)

What was the effect of adding line 25 to the program? (There was no effect.)

What is the use of the REM statement in line 25? (The REM statement was used to document a portion of the program to show any programmer the point at which the data was RESTORED to restart the data list.)

What is a REM statement? (A REM statement is a statement within a program that contains the programmer's remarks or comments and that in no way affects the output of the program. The REM statement has no effect on a program and will not change any values within the program.)

8. REM statements are used to document a program. In professional programs, a programmer often begins by stating the title and function of the program and the name of the company. Programmers' names and version numbers also often appear in REM statements.

Insert within the above program your name and a title for the program. Use a series of REM statements to accomplish this task.

Answer:

```

5 REM * CHANGE THE LETTERS*
6 REM ** THIS PROGRAM WILL SHOW THE USE OF
7 REM ***THE RESTORE COMMAND IN A PROGRAM
8 REM **** BY LORI ANN

```

9. Why are REM statements, or program documentation, necessary to users of a program?
 - . To help them to understand what is happening step-by-step as a program executes
 - . To alert other programmers to the purpose of each section of a program that they may be engaged in rewriting
 - . To see the relationship of each part of the program to the others
 - . To help future programmers make easier modification of program lines
 - . To help programmers understand the logic and sequence of a program's flow

Application:

1. Write a program with documentation that will store the names, addresses, and phone numbers of five friends in a data statement, ask for the birthday of each, and print out the name, address, and birthday of each.

Summary:

1. What is the purpose of the READ/DATA statement?
2. Why must a programmer be careful of sequence when more than one variable use data from a DATA statement?
3. How does an internal pointer handle sets of DATA values for multiple variables?
4. How may the same set of DATA be accessed more than once within a program without retyping the DATA?
5. What is the purpose of a REM statement within a program?

Assignments:

1. Find and correct any errors in each pair of READ/DATA statements:

```
10 READ A, A$, B
20 DATA 3,ED,9,
```

```
10 READ X$ Y$
20 DATA "HI" "ANN"
```

```
10 READ C, D
20 DATA 4,H,0
```

```
10 DATA A,5,4,78,1,3
20 READ A
```

```
10 READ X, Y$
20 DATA 5, Jump, 6, Jumped, 7, Jumping
```

2. Rewrite the following program using READ/DATA statements and document the program using REM statements. In using the DATA statement, make up five pieces of data for names and heights.

```
5 INPUT "What is your name"
10 INPUT "What is your height (in inches) ";H
20 Print "Your height in feet is"
30 F= H/12
40 PRINT F
50 GOTO 5
60 END
```

3. Give the output for the following programs:

```
10 DATA 8,10, 9, 12,26,14
20 READ A,B
30 REM *The sum of the first two numbers
40 C= A + B
50 PRINT C
```

```

60 RESTORE
70 READ A,B,C,D
80 REM**The sum of the first four numbers
90 D= A + B + C + D
100 PRINT D
110 RESTORE
120 READ A,B, C, D, E, F
130 REM***The sum of the first six numbers
140 E = A + B + C + D + E + F
150 PRINT E
160 END

```

```

10 X = 1
20 READ A$
30 PRINT X, A$
40 X = X + 1
50 GOTO 20
60 DATA SAM, ELLEN, ANDY, JACK, PAUL

```

4. Write a program to print the perimeter and area of rectangles if the lengths and widths are as follows:

LENGTH	WIDTH
12 cm	23 cm
9 m	14 m
62 cm	31 cm
5 cm	5 cm

Use READ/DATA statements as well as REM statements to document the program.

Appendix A8

BRANCHING (PART I): REVIEW

Performance Objectives: Students will be able to:

- . determine how comparisons are made during the execution of a program
- . use an IF/THEN statement to determine a response path
- . write programs using IF/THEN statements involving numeric variables and string variables
- . compare the ways a loop can be created by use of an IF/THEN statement in conjunction with a GOTO statement

Vocabulary: New: infinite loop

Review: conditional branching, decision statements, GOTO,
IF/THEN, INT

Motivation:

- . Forecast the output of the following program:

```
10 INPUT "What is your name ";N$
20 INPUT "What is your address ";A$
30 PRINT "HI ";N$;" MY NAME IS COMPUTE AND I LIVE IN A COMPUTER"
40 PRINT "I KNOW THAT YOU LIVE AT ";A$
50 PRINT
60 GOTO 10
70 END
```

- . Why will this program never stop? Why does it continue to ask for your name and address?
- . Suppose you wanted to put a special password into the program that would cause the execution of the program to end once this word was typed. How might this be accomplished?

Aim: How can we write a program using conditional branching?

Development:

1. Identify a password to be used within the program, for example, FINISHED, END, HELLO, GOODBYE, ZORBA. This password is to be used as the final entry when the computer asks for the name. Upon being entered into the computer, the computer will then GOTO line 70 and end the program.
2. How can this password concept be implemented? (IF N\$ = "FINISHED" THEN GOTO 70)
3. Explain the general form of the IF/THEN statement:

IF (condition) THEN GOTO (line #)
4. How does a computer check the condition? (If the response to the condition is yes, the computer goes to the line number indicated in the

THEN part of the statement. If the response to the condition is no, the computer simply executes the next numbered statement.)

5. Rewrite the program so that the computer will halt the program's execution when the word "GOODBYE" is typed in.

Acceptable answer:

```
10 INPUT "What is your name ";N$
25 IF N$ = "GOODBYE" THEN GOTO 70
20 INPUT "What is your address ";A$
30 PRINT "HI ";N$;" MY NAME IS COMPUTE AND I LIVE IN A COMPUTER"
40 PRINT "I KNOW THAT YOU LIVE AT ";A$
50 PRINT
60 GOTO 10
70 END
```

6. Review the fact that PRINTed strings must be enclosed within parentheses while numeric data may not be so enclosed.
7. Write a program that will examine the data below and decide if the person is a child (younger than 13), teenager (13 to 19 inclusive), or adult (older than 19).

The computer should output the person's name followed by the word "child", "teenager", or "adult". Store all data within a DATA statement.

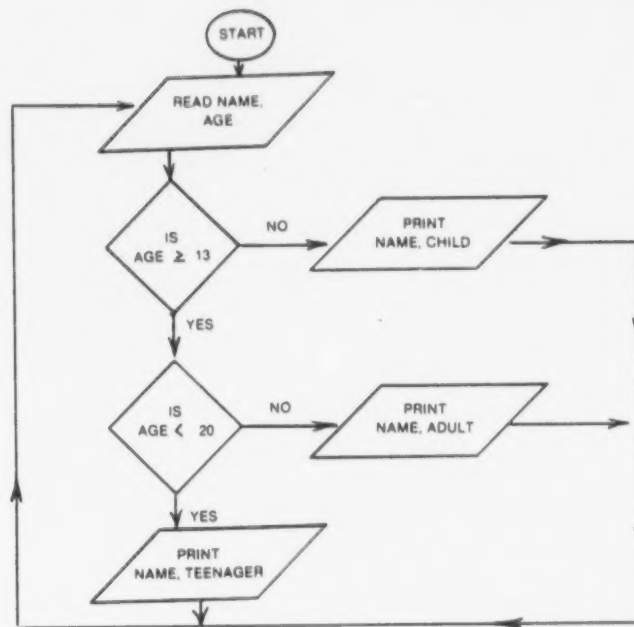
Acceptable answer:

```
10 READ A$, B
20 IF B<13 THEN GOTO 40
30 IF B>19 THEN GOTO 60
40 PRINT "TEENAGER"
50 GOTO 100
60 PRINT A$, "CHILD"
70 GOTO 100
80 PRINT A$, "ADULT"
90 GOTO 100
100 DATA ADAM, 14, BETSY, 30, CARL, 16, ED, 13, GINA, 20, LAUREN, 4
110 GOTO 10
120 END
```

8. How can we prevent the "OUT OF DATA" error that occurs when this program is RUN? Can we put a password, or flag, into the program so that the computer will recognize the program is ended with the last READING of the DATA statement?

Show that a flag is a type of trailing value within a DATA statement that the computer will recognize as a branch within a decision statement.

Review the flowchart for this program, including the decision statement to terminate the program when the flag is READ.



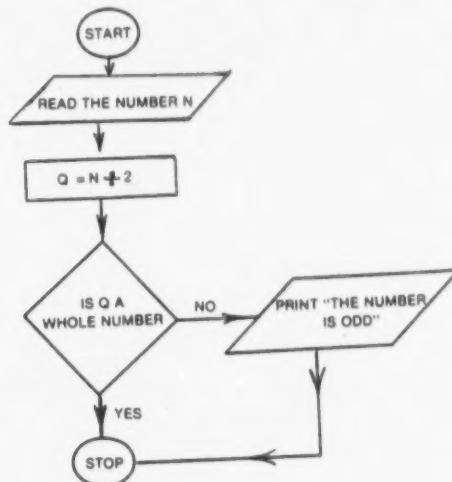
Applications:

1. Prepare a flowchart and write a program for the following problem:

We know that an even number is a whole number that is divisible by two. (It may be divided by two with a quotient that has no fractional part.) Any whole number that is not an even number is called an odd number.

Write a flowchart and program that will allow a number to be INPUT and then have the computer output the word EVEN or ODD.

Hint: The use of the INTEGER function may be helpful -- for example, $\text{INT}(2.6) = 2$, $\text{INT}(7) = 7$, $\text{INT}(12/5) = 2$.



Program:

```
10 INPUT "What number do you want tested ";X
20 IF X = -1 THEN GOTO 90
30 REM* TEST FOR DIVISIBILITY BY 2
40 D = X/2
50 I = INT(D)
60 IF I = D THEN PRINT "EVEN NUMBER"
70 IF I <> D THEN PRINT "ODD NUMBER"
80 GOTO 10
90 END
```

Summary:

1. How are a GOTO statement and an IF/THEN statement similar? (Both can be used to cause branching that will transfer the program flow in a particular direction.)
2. How are conditional branching and unconditional branching different?
3. How can comparisons of variables, either string or numeric, be made?

Assignments:

1. Which of the following are valid statements:

```
10 IF X>= 10 THEN GOTO 60
20 IF 10 < Z THEN GOTO 100
50 IF X > Y and Y>Z THEN GOTO 20
30 IF X = Y = 4 THEN GOTO 20
55 IF M$ >= HELLO THEN PRINT "GOODBYE"
```

2. Write a program that will use the following data via READ/DATA statements and will then determine the average salary per month of each employee. Use a flag within the program to prevent an OUT OF DATA error from occurring.

NAME	week 1	week 2	week 3	week 4
Louise Jay	\$65.52	\$120	\$58.50	\$30.00
Ed Cook	\$120	\$48.50	\$25.25	\$120
James Finn	\$15.55	\$89.89	\$90	\$100
Ann Luis	\$80.75	\$200	\$45	\$25.95

3. The varsity football team allows only those players who weigh at least 165 pounds to play ball. If a student weighs more than 140 pounds but less than 165 pounds that student is placed on the junior team. If the student is less than 140 pounds, team participation is not permitted.

Write a program that will output a list of those on the varsity team, another list of those on the junior team, and a third list of those not permitted to be on either team.

The list of students is as follows:

NAME	WEIGHT
Jackson	180 lbs
Lee	160 lbs
Mayers	200 lbs
James	108 lbs
Allison	176 lbs
Wright	128 lbs

Assignment for More Advanced Students:

1. Write a program that will alphabetize the following words: gras, tour, yellow, apple.

Appendix A9

BRANCHING (PART II): REVIEW

Performance Objectives: Students will be able to:

- . identify the different types of multiple-path choices
- . trace the progression of a program involving a multiple-path choice
- . write a program using a multiple-path choice to branch

Vocabulary: New: multiple branching, ON ... GOTO
Review: IF-THEN-ELSE, GOTO, >, <, >=, <=

Motivation:

- . Determine the output from RUNNING the following program:

```
10 READ A$,B
20 IF A$ = "END" THEN GOTO 70
30 IF B < 18 PRINT A$; " MINOR"
40 IF B >= 18 PRINT A$; " ADULT"
50 GOTO 10
60 DATA JED, 16, AMY, 12, LISA, 23, FRANCIS, 18, MARY, 35, END, 0
70 END
```

We know that when B had a value less than 18, the computer would output the word "MINOR" next to the person's name. What happened when B had a value of 18 or greater? (The computer printed the word "ADULT" next to the person's name.)

Could we simplify this program in any way?

Would it be possible to consolidate lines 30 and 40 into a single statement?

Aim: How can a multiple-path choice be created within a program?

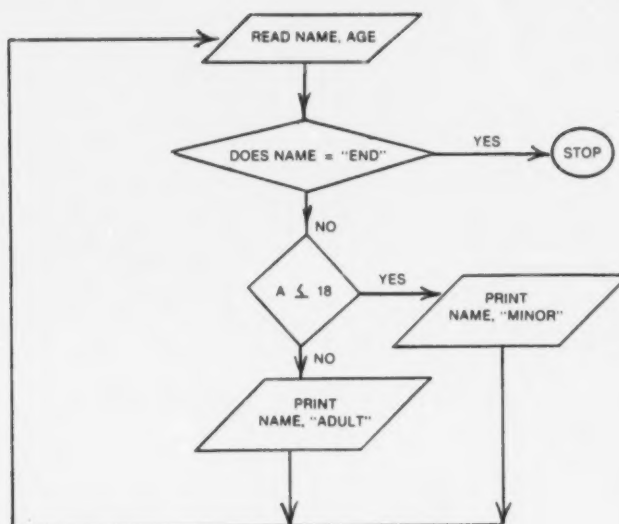
Development:

1. Draw a flowchart corresponding to the sample program.
2. Where does the branching occur? ($B < 18$, $B \geq 18$)

Why does the branch from $B \geq 18$ return the user to the decision statement $B < 18$? (If $B \geq 18$ is false, then $B < 18$ must be true.)

Review the inequality relators >, <, >=, <=

How can our two statements be consolidated into a single decision within the flow chart? Rewrite the flowchart incorporating this.



3. How can this single decision statement be translated into a programming statement? Introduce the use of the word ELSE within a decision statement (even if your computer cannot handle ELSE).

The ELSE statement may be incorporated within an IF/THEN statement as an alternative when the condition for the IF/THEN is not met.

Note: Some machines will not permit the ELSE choice and therefore this option is not always possible. Even if the school computers cannot implement this function it is an important concept to teach and should be covered with the class.

In the flowchart above, if $B < 18$ then the computer will print the word "MINOR". What is the alternative to $B < 18$? ($B \geq 18$)

What happens when $B \geq 18$? (The computer prints "ADULT".)

How can we translate that into BASIC in a single statement?

(IF $B < 18$ THEN PRINT "MINOR" ELSE PRINT "ADULT")

4. How can we consolidate the IF/THEN statements in the following program:

```

10 REM *This program will determine if a
20 REM **number is divisible by 5
30 INPUT " type in the number to be tested for divisibility "; N
40 X = N/5
50 IF N/5 = INT (N/5) THEN PRINT " DIVISIBLE"
60 IF N/5 <> INT(N/5) THEN PRINT "NOT DIVISIBLE"
70 GO TO 30
80 END
  
```

(50 IF $N/5 = \text{INT}(N/5)$ THEN PRINT "DIVISIBLE" ELSE PRINT "NOT DIVISIBLE")

5. The membership fees at the local YMCA are \$200 for one person and \$325 for a family (two or more persons). Write a program that will read a list of members' names and the number of persons within their groups and then will print the membership fees for the year for all members.

The data is:

NAME	NUMBER IN GROUP
LOGAN	6
FLYNN	2
GRAY	1
MORAN	4
SHORE	1

Acceptable answer:

```
10 READ A$,B
20 IF B>=2 THEN P = 325 ELSE P = 200
30 PRINT "DEAR ";A$;" YOU OWE $";P;" for YMCA membership"
40 DATA LOGAN, 6,FLYNN, 2,GRAY, 1,MORAN, 4,SHORE, 1
45 IF B>0 THEN GOTO 10
50 END
```

6. Now let us examine the following program:

```
10 REM* This program will determine the area
20 REM** of a square, triangle or circle
30 PRINT "What type of figure do you wish to determine the area of?"
40 INPUT" type the number corresponding to the figure - 1=square
   2=triangle 3=circle ";X
50 IF X = 1 THEN GOTO 80
60 IF X = 2 THEN GOTO 200
70 IF X = 3 THEN GOTO 300
80 INPUT" what is the length of the side?";S
90 A = S*S
100 PRINT "THE AREA OF THE SQUARE IS";A
110 GOTO 400
200 INPUT"what is the length of the base";B
210 INPUT"what is the length of the altitude";C
220 A = (B * C)/2
230 PRINT"THE AREA OF THE TRIANGLE IS";A
240 GOTO 400
300 INPUT "What is the length of the radius";R
310 A = R * R * 3.14
320 PRINT "THE AREA OF THE CIRCLE IS"; A
340 GOTO 400
400 END
```

How many branches exist within this program?

Show that the IF/THEN branching creates a multiple branch with three different possibilities.

Would it be possible to use a single IF/THEN ...ELSE statement? (only if

the ELSE is a branch to another IF/THEN statement, for example, IF X = 1 THEN GOTO 80 ELSE GOTO -----)

Is there a way of performing a multiple branch like this using a single statement?

7. Introduce the ON/GOTO statement as a means of accomplishing this task.

The ON/GOTO statement will permit you to combine several "equality type" IF/THEN statements into one.

By using the ON/GOTO statement the program branches to a specific program line.

ON X GOTO 80, 200, 300

ON (number) GOTO (line numbers)

The value of X determines to which line the program will branch.

In this case: If X is 1 then go to line 80
 If X is 2 then go to line 200
 If X is 3 then go to line 300

The value of X is always greater than 0.

8. What would happen if X = 4? (If there was a fourth line number in the list, the computer would GOTO that line. If there was no fourth line number the computer would then proceed to the next statement in the program just as it does when an IF's condition is not met.

Summary:

1. How can multiple branching from the same statement be accomplished via any number of routes?
2. How does the IF/THEN statement compare to the IF/THEN/ELSE statement and to the ON/GOTO statement?

Assignments:

1. Write a program that will take a specific temperature and convert it from Fahrenheit to Celsius or vice versa. The computer program should include an input of the temperature as well as an indication of whether it is Fahrenheit or Celsius.

Note: $F = (9/5) * C + 32$
 $C = (5/9) * (F - 32)$

2. Write a program that will input two numbers and then output the larger number. Use an IF/THEN/ELSE statement to accomplish this task.

3. If the value of B is 2, determine which line will receive control in each statement below.

ON B GOTO 20, 44, 50, 100
ON B GOTO 60, 80, 40, 75, 100
ON B GOTO 50, 10, 20
ON B GOTO 5, -1, 10, 50, 100, 20

4. Write a program that will ask for your favorite number from 1 to 5 and then print a secret message of your choice based on the number you chose. Use an ON/GOTO statement to accomplish this task.

Appendix A10

FOR/NEXT LOOPS: REVIEW

Performance Objectives: Students will be able to:

- . define the function of a FOR/NEXT loop
- . use a FOR/NEXT loop within a program
- . use a FOR/NEXT loop as a means of counting
- . use a FOR/NEXT loop to provide time delay within a program

Vocabulary: New: STEP

Review: accumulator, FOR/NEXT loop

Motivation:

- . Examine the following program, trace its path, and predict its output:

```
10 X = X + 1
20 PRINT X;" HELLO"
30 IF X = 6 THEN GOTO 50
40 GOTO 10
50 END
```

What would the output have been if line 30 had been omitted?

What happens in line 40?

What are the lines between 10 and 40 collectively called?

What occurs within a loop?

What is the function of line 10?

How might the original program been revised so that the message would be printed 25 times?

How else might we have written this program so that the computer would execute the same set of instructions a specific number of times?

Aim: How might we use the FOR/NEXT loop to cause a computer to execute a specific set of instructions a given number of times?

Development:

1. What occurs within the loop in the Motivation program? What program lines will be repeated for a specific number of times? (lines 10 to 40)

How many times will this loop be executed? (six times)

What variable is used to count from 1 to 6? (X)

2. Now let us examine the following program, which will produce the same outcome:

```

10 FOR X = 1 TO 6
20 PRINT X;" HELLO"
30 NEXT X
40 END

```

What does line 10 do? (Line 10 tells the computer that a loop must be executed six times.)

What does line 30 do? (Line 30 tells the computer to go back to line 10, verify that the loop has been performed, and then perform it again if necessary. After it has performed the sixth series, the computer will then advance to line 40, where it will END. Since most computers increment their variable counter when they reach the NEXT statement and only check to see if they should proceed with the loop after incrementing the variable, the counting variable will always be one too large at the end of the looping.)

3. The FOR/NEXT loop tells the computer to do something for a specific number of times. The computer then repeats anything placed between the words FOR and NEXT the requested number of times.

In a FOR/NEXT loop there is no need to initialize a value for the counting variable because the computer automatically initializes the value of the variable to the starting value found in the FOR statement.

In the following FOR statements, what is the initial value of the variable?

```

10 FOR X = 6 TO 100      (6)
20 FOR B = 0 TO 4        (0)
30 FOR M = 200 TO 1000   (200)

```

4. Predict the outcome of the following program:

```

10 PRINT"This program is an adding program"
20 FOR I = 1 TO 10
30 S = S + I
40 NEXT I
50 PRINT "The sum of the numbers is ";S
60 END

```

(The sum of the numbers is 55)

What happened in this program? (The sum of the first 10 whole numbers was determined.)

What was the purpose of line 30? (Line 30 acted as an accumulator, adding each new number to the sum already calculated: new sum = old sum + I)

Modify this program so that we can see the sum accumulating as well as the number that is being added.

Acceptable answer:

```
10 PRINT "This program is an adding program"
20 FOR I = 1 TO 10
30 S = S + I
40 PRINT I, S
50 NEXT I
60 END
```

5. Write a program that will print the whole numbers from 100 to 200, using a FOR/NEXT loop.

Acceptable answer:

```
10 FOR A = 100 TO 200
20 PRINT A
30 NEXT A
40 END
```

Stress the speed of execution with which this task is accomplished by the computer. The execution is so fast that it is almost impossible to read all the numbers as they are being printed out.

How might we slow down the output of this program so that we may easily read the numbers as they are printed out? (Insert a pause.)

Where would this pause be inserted? (somewhere between lines 10 and 30, either before or after line 20)

Would it be possible to use a FOR/NEXT loop to accomplish this task? (yes)

6. Have a student count aloud from 1 to 10. Then, have the student count from 1 to 5 silently, adding a silent pause after each of those numbers, before continuing to count from 6 to 10 aloud.

What occurred? (A pause was inserted after each of the first five numbers was counted silently.)

How can the computer count silently before it outputs the next number? Is it possible to use a FOR/NEXT loop? (yes)

How might we use a FOR/NEXT loop for this task?

Acceptable answer:

```
25 FOR I = 1 TO 5
28 NEXT I
```

Show that the pause causes the computer to internally count from 1 to 5 before continuing past line 28.

How might we make the pause longer? (By increasing the largest value of

I from 5 to something larger. The larger the value, the longer the pause.)

7. Write a program that will PRINT the even numbers from 40 to 200, using a pause so that we can easily read the output as it is printed on the screen.

Acceptable answer:

```
10 FOR A = 40 TO 200
20 PRINT B
30 FOR I = 1 TO 25
40 NEXT I
50 PRINT
60 NEXT A
70 END
```

Now, type in the following program and compare the results with the results from the last program:

```
10 FOR A = 40 TO 200 STEP 2
20 PRINT A
30 FOR I = 1 TO 25
40 NEXT I
50 PRINT
60 NEXT A
70 END
```

Show that this program will give the same output as the previous one did.

What happens in line 10? (The computer is instructed to perform some instruction 40 to 200 times, in increments of 2.)

The command STEP--- within a FOR statement instructs the computer to count in increments equal to the step. Normally computers STEP by 1.

How would the computer count in the following program lines?

```
10 FOR X = 1 TO 100 STEP 10 (10,20,30,...100)
20 FOR T = 1 TO 500 STEP 50 (1, 51, 101, 151,...)
30 FOR A = 100 TO 1000 STEP 200 (100,300,500...)
40 FOR Z = 65 TO 100 STEP 5 (65,70,75,...100)
```

Applications:

1. Predict the outcome of the following programs:

```
10 A$ = "I REPEAT"
20 FOR I = 1 TO 100
30 PRINT I, A$
40 PRINT
50 NEXT I
60 FOR J = 1 TO 5
70 PRINT "THE END"
```

```

80 NEXT J
90 PRINT "IS HERE!!!"
100 END

10 PRINT "N", "N x N" , "N x N x N"
20 For N = 1 TO 10
30 PRINT N,N*N,N*N*N
40 NEXT N
50 END

```

2. Write a program that will print three stars in a column, pause, clear the screen and begin again.

Summary:

1. What three statements can be used to create a loop within a program? (IF/THEN, GOTO, FOR/NEXT)
2. How can a FOR/NEXT statement loop create a time delay?
3. What is the purpose of including the STEP statement within a FOR/NEXT loop?

Assignments:

1. Write a program that will add all even numbers to 100 and then print the sum.
2. Write a program that will ask you to input a message and will ask the number of times you would like the message printed out, and will then follow your additional instructions. Use INPUT statements as well as FOR/NEXT loops.
3. Create a program that will output the following pattern ten times, with a long pause between outputs.

```

  *   *   *   *
    *   *   *
      *   *
        *

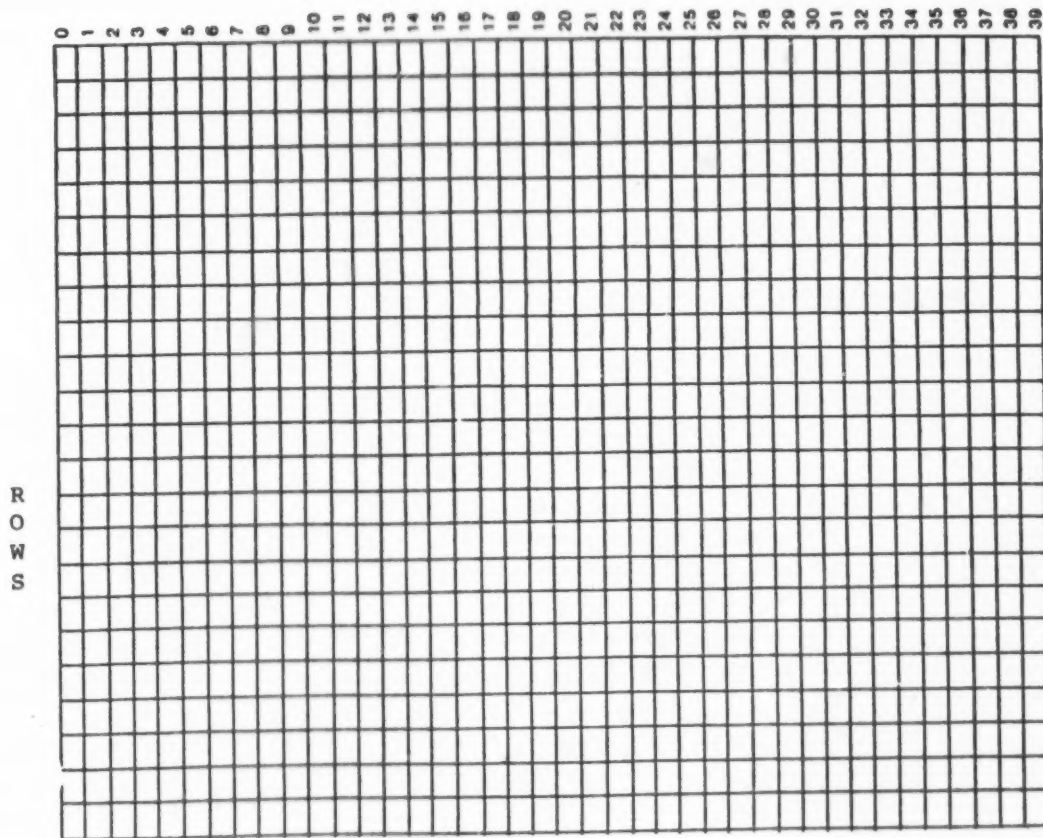
```

4. Write a program that will ask for a student's name and the number of grades to be added. The computer must output the student's name, grades, and average of the grades.

Use INPUT statements as well as a FOR/NEXT loop to accomplish this task. When the student's data is output, have the computer ask if there are more students and then, according to that response, either return to the beginning for the next student or terminate the program.

VIDEO DISPLAY SCREENS

COLUMNS



APPLE 80 COLUMNS x 24 ROWS

COLUMNS

0 1 2 3 4 5 6 7

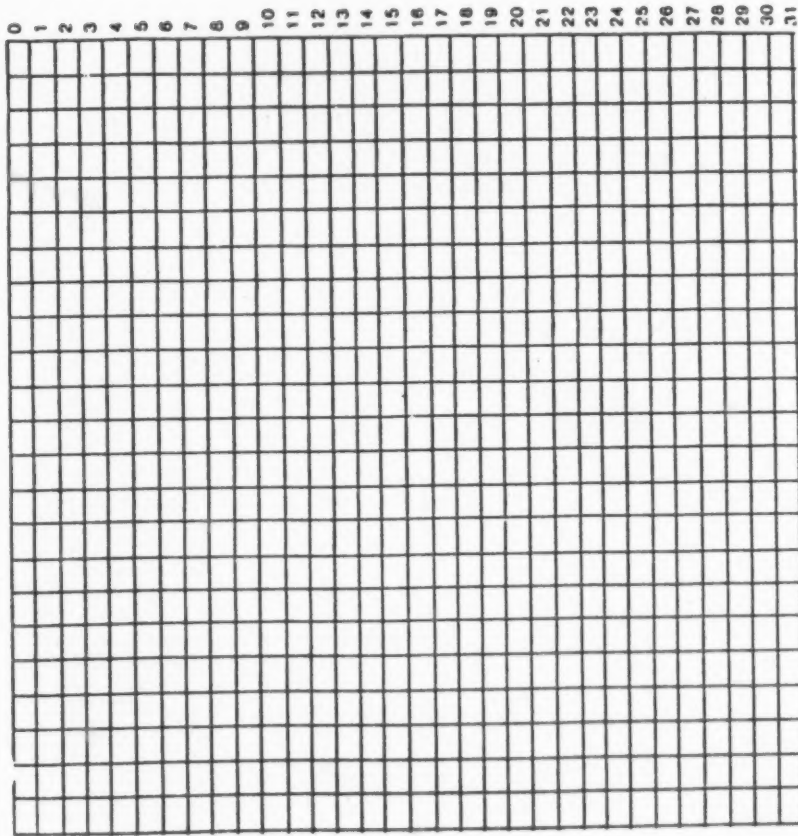
73 74 75 76 77 78 79

R
O
W
S

COMMODORE - 64 32 COLUMNS x 24 ROWS

COLUMNS

R
O
W
S



ATARI 40 COLUMNS x 20 ROWS

COLUMNS

ROWS

[illegible]

IBM - PC/PC JR 40 COLUMNS x 25 ROWS

COLUMNS

ROWS

This is a full-page view of a blank sheet of graph paper. It features a uniform grid of small squares. Along the top edge, there are numerical labels from 0 to 39, spaced evenly. Along the left edge, there are alphabetical labels from A to J, also spaced evenly. The rest of the page is an empty grid area.

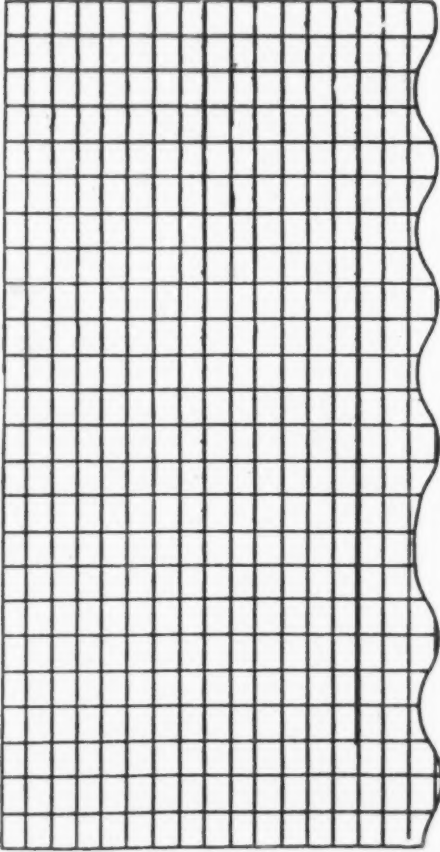
IBM - PC/PC JR

80 COLUMNS x 25 ROWS

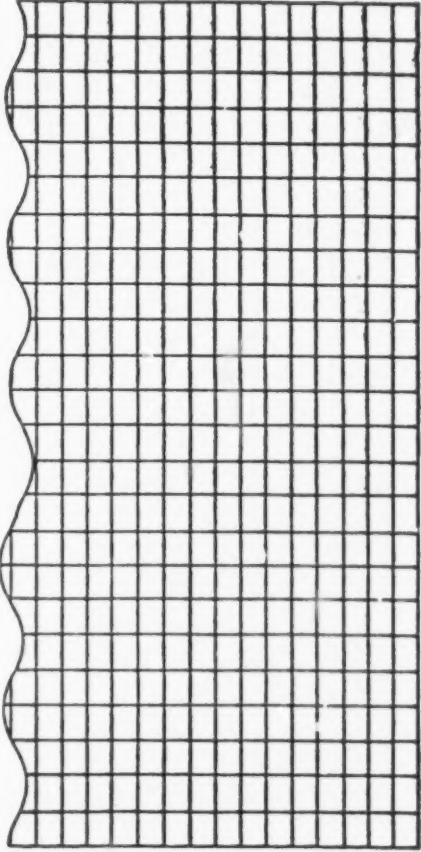
COLUMNS

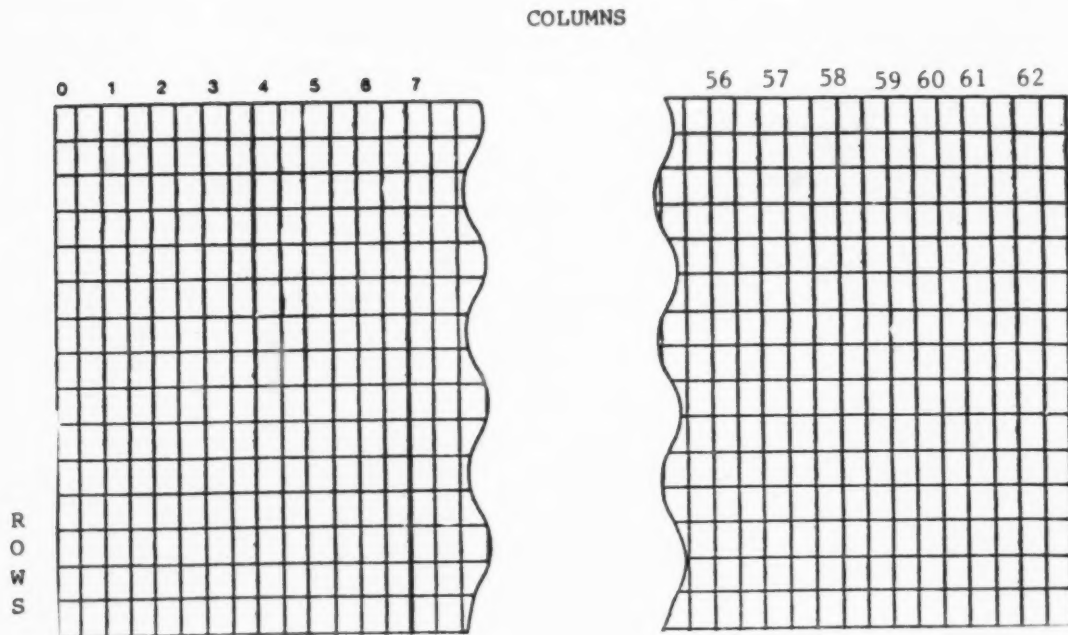
0 1 2 3 4 5 6 7

R
O
W
S



73 74 75 76 77 78 79





COLUMNS

R O W S	0	1	2	3	4	5	6	7		73	74	75	76	77	78	79